НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»
на тему: «Комп'ютерна гра
на базі рушія Unreal Engine 4»

Виконав:
студент IV курсу, групи ІО-63
Маслачков Олексій Юрійович _____

Керівник:
Старший викладач
Сімоненко Андрій Валерійович _____

Консультант з нормоконтролю:
Професор, доктор технічних наук
Сімоненко Валерій Павлович _____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ім'я, по батькові _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Маслачкову Олексію Юрійовичу

1. Тема проєкту «Комп'ютерна гра на базі рушія Unreal Engine 4», керівник проєкту Сімоненко Андрій Валерійович, старший викладач, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту 06 червня 2020 р.

3. Вихідні дані до проєкту: технічне завдання, науково-технічна література

4. Зміст пояснювальної записки: порівняльний аналіз існуючих програмних рішень, вибір засобів реалізації та опис отриманої системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) :

1. Функціональна схема – плакат
2. Принципова схема – плакат
3. Структурна схема – плакат

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П., професор		

7. Дата видачі завдання 01 вересня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	01.09.2019-22.12.2019	
2	Вивчення та аналіз завдання	23.12.2019-20.03.2020	
3	Розробка архітектури та загальної структури системи	20.03.2020-01.04.2020	
4	Розробка структур окремих підсистем	01.04.2020-10.04.2020	
5	Програмна реалізація системи	11.04.2020-20.04.2020	
6	Оформлення пояснювальної записки	01.05.2020-23.05.2020	
7	Передзахист	24.05.2020-26.05.2020	
8	Захист	15.06.2020-20.06.2020	

Студент Олексій МАСЛАЧКОВ

Керівник Андрій СІМОНЕНКО

АНОТАЦІЯ

Дана дипломна робота присвячена розробці комп’ютерної гри на базі рушія Unreal Engine 4, що призначена для проведення дозвілля.

Гравцю потрібно вижити у змаганні проти великої кількості інших героїв, що контролюються штучним інтелектом.

Для реалізації гри використовується ігровий рушій Unreal Engine 4 та Blueprints - система візуального скриптинга.

ANNOTATION

This thesis is devoted to the development of a computer game on the basis of the Unreal Engine 4 game engine, which is designed for leisure.

The player must survive in a competition against a large number of other heroes controlled by artificial intelligence.

To implement the game, the game engine Unreal Engine 4 and Blueprints (a system of visual scripting) are used.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	*ІАЛЦ. 467800.001* ВП	Відомість проєкту	1	
3	А4	*ІАЛЦ. 467800.002* ТЗ	Технічне завдання	3	
4	А4	*ІАЛЦ. 467800.003* ПЗ	Пояснювальна записка	81	
5	А4	*ІАЛЦ. 467800.004* Д1	Функціональна схема	1	
6	А4	*ІАЛЦ. 467800.005* Д2	Принципова схема	1	
7	А4	*ІАЛЦ. 467800.006* Д3	Структурна схема	1	

				ІАЛЦ. 467800.001 ВП	
	ПІБ	Підп.	Дата		
Розробн.	Маслачков О.Ю.			Відомість дипломного проєкту	Лист 1 / Листів 1
Керівн.	Сімоненко А.В.				
Консульт.					КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-63
Н/контр.	Сімоненко В.П.				
Зав.каф.	Стіренко С.Г.				

Технічне завдання

до дипломного проєкту

на тему: «Комп’ютерна гра

на базі рушія Unreal Engine 4»

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку програмного додатку на тему «Комп’ютерна гра на базі рушія Unreal Engine 4».

Область застосування: організація дозвілля та розвиток різноманітних навичок гравця.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка ігрового додатку, що слугує для організації дозвілля та розвитку різноманітних навичок.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є технічна документація для інтерфейсів прикладного програмування та бібліотек, що використовуються під час розробки програмного продукту, науково-технічна література з інформаційних технологій, публікації в періодичних виданнях, а також відповідні статті в мережі Інтернет за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Зберігання даних на пристрої.
- Конкурентоспроможність ігрового додатку на ринку комп’ютерних ігор.
- Зручність і простота графічного інтерфейсу розроблюваної системи.

5.2. Вимоги до програмного забезпечення

- Система візуального скриптинга Blueprints.
- Використання ігрового рушія Unreal Engine 4.

5.3. Вимоги до апаратного забезпечення

- Комп’ютер на базі чотирьохядерного процесору частотою від 2.5 ГГц
- Відеокарта NVIDIA GeForce 470 GTX або AMD Radeon 6870 HD і вище
- Оперативної пам’яті не менше 8192 Мбайт.
- Підключення до мережі Інтернет.

Пояснювальна записка

до дипломного проєкту

на тему: «Комп’ютерна гра

на базі рушія Unreal Engine 4»

Київ – 2020 року

ЗМІСТ

ВСТУП

Індустрія комп'ютерних ігор з'явилася відносно недавно, в середині 1970-х років, але вже змогла розвинутися в величезну галузь з колосальними доходами в кілька мільярдів доларів на рік. Зрозуміти подібне раптове зростання популярності віртуальних розваг дуже просто: все це завдяки широкому розповсюдженню комп'ютерних технологій, в тому числі завдяки появі мережі Інтернет. Завдяки цьому, на відміну від інших видів розваг, комп'ютерні ігри більш доступні для кінцевого користувача.

Для того, щоб просто пограти, гравцеві потрібно мати тільки комп'ютер або ігрову приставку і копію самої гри, а завдяки широкому розповсюдженню мережі Інтернет отримати копію гри можливо навіть не виходячи з дому. Більш того, споживачеві не потрібно мати особливих знань для того, щоб обрати собі гру, в той час як для більшості інших видів розваг необхідно розібратися як мінімум в необхідній екіпіровці.

Так само варто взяти до уваги, що комп'ютерні ігри останнім часом перестали позиціонуватися як програми тільки для відпочинку і розваг. Наприклад, сьогодні, завдяки використанню ігрових технологій, створюються спеціальні комплекси по симуляції, які служать для навчання та тренування фахівців в різних областях: від лісорубів до пілотів реактивних літаків.

В Україні, на жаль, все трохи інакше. Ігрова індустрія в нас, так само як і в більшості інших країн пострадянського простору, розвинена погано. Пов'язано це з тим, що культура комп'ютерних розваг прийшла до нас занадто пізно і практично не розвивалася. Через це, навіть при досить високому попиті, ми маємо дуже малу кількість компаній-розробників, які можуть скласти конкуренцію закордонним компаніям.

Найбільш відомими українськими компаніями з розробки комп'ютерних ігор є 4A Games і GSC Game World.

Перша прославилася на весь світ дуже популярною серією ігор під назвою "Metro", при цьому на даний момент випущено вже чотири гри, в які із задоволенням грають люди з усього світу:

- Metro 2033;
- Metro: Last Light;
- Metro Redux;
- Metro Exodus.

Друга ж компанія, GSC Game World, спочатку була відома як творець серії ігор "Козаки", одних з кращих ігор у жанрі стратегія в реальному часі. Потім ця компанія стала відома всьому світу, коли в 2007 році була випущена гра "S.T.A.L.K.E.R.: Тінь Чорнобиля". Дія гри відбувається в зоні відчуження Чорнобильської АЕС, тому вона швидко стала культовою в країнах пострадянського простору, особливо в Україні та Росії. Пізніше були випущені ще дві гри, а саме "S.T.A.L.K.E.R.: Чисте небо" та "S.T.A.L.K.E.R.: Поклик Прип'яті", які увійшли в серію ігор "S.T.A.L.K.E.R.". Всі три гри виявилися настільки продуманими і динамічними шутерами, що стали дуже популярними навіть в країнах далекого зарубіжжя.

Звичайно існують і інші українські розробники комп'ютерних ігор, але їм не вдалося залучити до своїх ігор велику кількість гравців.

Також в Україні знаходяться дві з сорока дев'яти студій Ubisoft Entertainment - всесвітньо відомої французької компанії, яка спеціалізується на розробці та виданні комп'ютерних ігор і є одним з найбільших ігрових видавців у Європі. Дані студії розташовані в Києві і в Одесі. У них працюють українські фахівці, які приймають повну участь в розробці великих ігрових проектів на ряду з іншими сорока семи студіями, які розташовуються більш, ніж в двадцяти країнах.

Розвиток технологій в даному напрямку можна вважати одним з найбільш перспективних, особливо в нашій країні.

Об'єктом дослідження є розробка комп'ютерних ігор.

Предмет дослідження: технології розробки комп'ютерної гри.

Мета випускної кваліфікаційної роботи - розробити комп'ютерну гру на базі рушія Unreal Engine 4.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1) вивчити особливості і стан комп'ютерної індустрії України;
2) вибрати жанр, вид і платформу для комп'ютерної гри;
3) розробити сценарій, концепцію основних елементів;
4) вибрати і вивчити засіб реалізації;
5) підготувати необхідні для гри анімації;
6) реалізація комп'ютерної гри.

РОЗДІЛ 1

1. ДОСЛІДЖЕННЯ ПОНЯТТЯ «КОМП'ЮТЕРНА ГРА», ОГЛЯД РІЗНОМАНІТНИХ ТЕХНОЛОГІЙ РОЗРОБКИ КОМП'ЮТЕРНИХ ІГОР ТА ВИБІР КОНКРЕТНОЇ ТЕХНОЛОГІЇ ДЛЯ РЕАЛІЗАЦІЇ ПРОЕКТУ

1.1. Дослідження поняття і класифікація комп'ютерних ігор

Терміном комп'ютерна гра (іноді використовують термін відеогра) позначається інтерактивне програмне забезпечення, в якому організовано так званий геймплей (ігровий процес) та яке спілкується з користувачем, використовуючи двовимірну або тривимірну графіку та звуки. Таке програмне забезпечення використовується для розважальних чи освітніх цілей, працює на комп'ютері, смартфоні або на спеціалізованому комп'ютері, який називається ігровою консоллю та розроблений майже виключно для ігор та мультимедійних розваг.

Комп'ютерні ігри нерідко створюються на основі сторонніх джерел, таких як фільми, книги або комікси. Наприклад, є ігри, які створені за мотивами таких відомих фільмів, як "Зоряні Війни", "Матриця" та "Божевільний Макс". Також часто розробляють комп'ютерні ігри за мотивами книг. Найвідомішою серед них є серія ігор "The Witcher", яка створена за творами Анджея Сапковського. Неменш відомими є серії ігор "Metro" та "S.T.A.L.K.E.R.", які створені за романом Дмитра Глухівського та повістью братів Стругацьких відповідно. Любителі коміксів про супергероїв теж не залишилися осторонь, створюються ігри і з їх улюбленими персонажами в ролі головних героїв. Найвідомішою вважається серія відеоігор Batman: Arkham, в якій можна відчути себе самим Бетменом.

Деякі комп'ютерні ігри стають настільки популярними, що буває і навпаки, за їх мотивами знімають фільми. Так сталося з такими серіями ігор як "Resident Evil", "Tomb Raider", "Hitman", "Assassin`s Creed", "Prince of

Persia” та іншими. Існують комп’ютерні ігри не тільки для розважальних цілей, а й ті, які спеціально розроблені, щоб виступати в якості навчального матеріалу або дозволяти використовувати гравців в науково-дослідних цілях. Але такі ігри рідко випускаються в широкі маси.

За деякими з комп’ютерних ігор проводяться аматорські і професійні спортивні змагання, все це офіційно називається кіберспортом. Зазвичай змагання проводяться в спортивних симуляторах, шутерах або в стратегіях реального часу. Дисциплінами кіберспорту є конкретні ігри. На даний момент дисциплін дуже багато, але найпопулярнішими вважаються “Dota 2” та “Counter-Strike: Global Offensive”, саме з них проводяться наймасштабніші змагання, на яких збираються цілі стадіони глядачів.

Комп'ютерні ігри надали настільки істотний вплив на суспільство, що останнім часом в інформаційних технологіях з'явилася стійка тенденція до так званої гейміфікації (використання ігрових підходів, які широко поширені в комп'ютерних іграх) для неігрового прикладного програмного забезпечення. Наприклад, для потреб армій стали створювати спеціальні симулятори, які допомагають у тренуванні солдатів. Також створюються симулятори і для тренування представників інших важких професій, таких як льотчик, космонавт або автогонщик.

Цікавий факт, що комп'ютерні ігри з 2011 року офіційно урядом США і американським Національним фондом визнані окремим видом мистецтва, поряд з театром і кіно.

З усього цього випливає, що відеоігри щільно влилися в наше нинішнє життя. Більш того, сфера їх використання за останні 10 років сильно зросла, тепер ігри використовують не тільки для відпочинку та розваги, а також для навчання людей і проведення наукових досліджень.

Комп'ютерні ігри класифікують за кількома основними ознаками:

- жанр;
- кількість гравців;
- візуальне подання;

• платформа.

Жанр відеоігри визначається основною механікою гри, тобто методом взаємодії, що змінює стан ігрового середовища, а також її геймплеєм, ігровим процесом. Всього існує близько вісімдесяти жанрів комп'ютерних ігор, в таблиці 1 представлені найосновніші

Жанр	Основні піджанри	Особливості
Аркада (Arcade)	• Платформер • Файтинг • Скролер	Ігри з навмисне примітивним ігровим процесом.
Пригодницька гра (Adventure)	• Квест • Візуальна новела	Головною частиною гри є історія.
Рольова гра (RPG)	• Тактична • Екшен RPG • Японска (Західна) • ZPG	Гравець практично не обмежений у виборі ігрових предметів, напарників і діалогів.
Екшен (Action)	• Beat'em Up • Шутер • Слешер	Потребується часте і активне натискання кнопок управління.
Стратегічна гра (Strategy)	• Покрокова • У реальному часі • Глобальна • Варгейм • Економічна	Основа гри - необхідність гравцеві робити нетривіальний вибір.
Комп'ютерний симулятор (Simulator)	• Технічний • Симулятор життя	Гравець робить безліч вправ, відточуючи свою техніку.
Головоломка (Puzzle)		Вимагає аналітичного мислення.
Навчальна гра (Educational)		Під час виконання будь-яких дій в грі гравець навчається.

Представлені тут жанри не можна назвати усіма існуючими, так як останнім часом стали з'являтися ігри з власним жанром, який можна віднести як до одного з представлених тут жанрів, так і до самостійного окремо від них.

За кількістю гравців ігри поділяються на два види:

• однокористувацькі - розраховані тільки на одного гравця;

• багатокористувацькі - розраховані на велику кількість гравців.

За візуальним поданням комп'ютерні ігри можна розділити на наступні види:

• текстові - мінімальне графічне представлення, спілкування з гравцем проходить за допомогою тексту;

• 2D – всі елементи намальовані у вигляді двовимірної графіки (спрайтів);

• 2,5D – імітація тривимірного ігрового простору, однак при цьому графіка не є тривимірною;

• 3D – всі елементи намальовані у вигляді тривимірної графіки (3D - моделі).

Комп'ютерні ігри можуть розроблятися під такі платформи:

• персональні комп'ютери;

• ігрові приставки / консолі;

• мобільні телефони.

Представлена тут класифікація насправді не є повною і може бути доповнена. Так, наприклад, до жанрів можна додати такий специфічний вид ігор як музичні комп’ютерні ігри, де основна увага приділяється музиці або звукам. Багатокористувацькі ігри в свою чергу діляться на кілька підкатегорій. Але варто розуміти, що цієї класифікації досить для визначення більшості з існуючих відеоігор, так як на сьогоднішній день не існує однозначної і повної класифікації для комп'ютерних ігор. Це відбувається через те, що багато ігор не можна віднести до будь-яких критеріїв. Наприклад, гра може відповідати відразу кільком жанрам, вийти відразу на

декількох платформах або бути розрахованою як на одного гравця, так і на велику їх кількість.

1.2. Загальний алгоритм реалізації проекту

Загальний алгоритм розробки комп'ютерної гри мало чим відрізняється від алгоритму розробки будь-якого іншого програмного продукту. Він включає в себе три великих етапи:

1) проектування;
2) розробка;
3) випуск та підтримка.

На етапі проектування визначається, якою повинна бути гра, особливості її ігрового процесу, ігрового середовища. Також на цьому етапі визначаються засоби її розробки.

При визначенні особливостей комп’ютерної гри виділяються ідея, жанр та її сетинґ. Ідея - це те, що буде спонукати гравця грати в створювану гру. Вона дуже тісно пов'язана з жанром. Так, наприклад, основна ідея рольової гри (RPG) - дозволити гравцеві прожити свою роль так, як йому захочеться, а основна ідея шутера - дозволити гравцеві взяти участь в реальних чи вигаданих бойових діях. Таким чином виходить, що жанр буде підібраний практично відразу після визначення основної ідеї гри.

Після визначення ідеї та жанру відеогри, наступним кроком буде вибір сетинґу. Сетинґ - це середовище, в якому буде відбуватися основна дія гри. Він визначає місце, час і умови дії. Вибір сетинґу може сильно полегшити написання сценарію для гри, тому його краще вибирати заздалегідь, ґрунтуючись на смаки цільової аудиторії.

До засобів розробки в першу чергу відносять програмний код та ігровий рушій. Від їх грамотного вибору залежать як швидкість самої розробки, так і працездатність самого продукту надалі. Програмний код в першу чергу залежить від платформи, для якої буде створюватися комп'ютерна гра. Наприклад, якщо гра створюється для браузерів, то

доречним буде використання мови Java або Flash. Якщо відеогра створюється для персонального комп'ютера, оптимальним вибором буде, наприклад, мова програмування C++.

Ігровий рушій відповідає за низькорівневий опис фізики об'єктів, правил рендеринга графіки та ін. При виборі ігрового рушія насамперед дивляться на його доступність і вже зроблений вибір мови програмування. Наприклад, ігровий рушій Unreal Engine 4 дозволяє розробляти ігри на мові C++. Він є безкоштовним, але якщо щоквартальний прибуток компанії розробника перевищує 3000 доларів, то необхідно передавати компанії Epic Games (розробники ігрового рушія Unreal Engine) п’ять відсотків від прибутку з продажів гри.

Після визначення особливостей компьютерної гри і засобів розробки починається другий етап реалізації проекту - розробка.

Розробка - найбільший та найдовший етап реалізації проекту. Він включає в себе велику кількість кроків, без яких неможливо створити працездатний продукт.

Насамперед потрібно визначитися з сюжетом та ігровою механікою. Ігрова механіка ґрунтується на особливостях гри, вона визначає всі об'єкти та правила, за якими гравець буде взаємодіяти з ними.

Зазвичай паралельно з розробкою ігрової механіки йде написання сюжету гри. Сюжет грає не останню роль, він визначає те, наскільки буде гравцеві цікаво грати в вашу відеогру. Сюжет представляють в двох варіантах: літературний та режисерський сценарій. Літературний сценарій описує основні події та персонажів гри, які беруть участь в грі. Режисерський ж являє собою докладний опис рівнів гри, подій, які на цих рівнях відбуваються.

Так само на даному етапі починається раннє опрацювання графічної складової та дизайну гри. На основі сюжету і заздалегідь обумовленого дизайну створюються ранні концепт-арти (малюнки, які візуально передають

ідею майбутньої комп’ютерної гри), на основі яких згодом буде опрацьовано основний вид гри та її персонажів.

Після написання сюжету та розробки ігрової механіки починається найважливіша частина - розробка самої гри.

На основі сюжету і концепт-артів починається створення персонажів та об'єктів гри. Паралельно з цим йде розробка ігрових рівнів. При розробці ігрового рівня спочатку створюється його спрощений план, на якому схематично зображено сам рівень, а також зображені предмети, з якими згодом буде взаємодіяти гравець.

Слідом після цього створюється перша версія рівня. Зазвичай, вона являє собою просто голу локацію, з мінімумом необхідних для проходження предметів. Ця версія рівня служить для того, що б протестувати рівень на прохідність. Після тесту, рівень починають поступово заповнювати іншими об'єктами.

Незабаром після створення перших рівнів триває складання першого прототипу відеогри, який називають альфа-версією гри. Вона необхідна для того, що б розробник міг провести тестування (альфа-тестування) основної механіки гри та перевірити, наскільки вона відповідає заявленим вимогам. Часто в альфа-версії гри у об'єктів навіть немає текстур або вони взагалі представлені у вигляді абстрактних об'єктів.

Якщо альфа-версія гри успішно проходить тестування, настає наступний етап розробки - опрацювання механіки та об'єктів відеогри.

На даному етапі йде доробка рівнів і механіки гри, також починають додавати перші сюжетні події в гру, такі як відеоролики, сюжетні діалоги та кат-сцени (внутрішньоігрові відео). Так само виправляються перші помилки та несправності в коді відеогри, які були виявлені при тестуванні альфа-версії гри.

Після цього настає етап створення другого прототипу гри або, як прийнято говорити, бета-версії. Бета-версія служить для того, що протестувати відеогру на несправності, фактично бета-версія являє собою

практично готову гру. У ній можуть бути відсутні які-небудь незначні елементи, які не впливають на геймплей гри. При тестуванні бета-версії гри перевіряється абсолютно все. Часто, особливо, якщо гра є багатокористувацькою, на тестування запрошуються звичайні гравці - це дозволяє сильно прискорити час проведення тестування, а так само зняти частину навантаження з команди розробки.

Якщо гра проходить бета-тестування, вона відправляється на остаточне доопрацювання та виправлення критичних помилок, після чого йде збірка фінальної версії гри, а потім слідом настає реліз (випуск) гри.

Після релізу гри продовжується подальша її підтримка. Підтримка полягає у випуску патчів (файлів виправлень помилок в готовому продукті). Так само для того, що б продовжити життєвий цикл гри, для неї випускають додатковий контент у вигляді так званих DLC (Downloadable Content), які додають різні предмети і / або можливості для гравця.

Таким чином, стало зрозуміло, що етапи розробки комп'ютерних ігор мало відрізняються від етапів розробки будь-якого іншого програмного продукту.

1.3. Аналіз засобів створення комп’ютерних ігор

Основною складовою будь-якої комп'ютерної гри є її рушій - це її основне ядро, базове програмне забезпечення, на основі якого будуються всі інші складові гри. Програмний код, який може використовуватися для створення варіацій гри, доповнення до неї або навіть зовсім нового ігрового світу.

Вперше визначення з'явилося в середині 90-тих років, коли почали з'являтися ігри, схожі на головний шутер того часу - Doom. У той же час у вільному доступі почали з'являтися ігрові рушії, на основі яких і сторонні розробники, і звичайні користувачі могли пробувати розробляти власні ігри.

З тих пір ігрові рушії ставали все більш складними технічно, довгими і насиченими за своїм програмним кодом. Але при цьому, як і на початку

свого існування, вони містять в собі жорстко фіксовані дані: ігрову логіку, фізику об’єктів, правила відтворення об’єктів.

На основі рушія прописуються всі інші складові гри, і їх чимало. Тому навіть при використанні одного і того ж рушія в результаті віртуальні світи виходять абсолютно різними.

Отже, головним засобом для розробки комп'ютерної гри є ігровий рушій. Ігрові рушії відразу включають в себе всі необхідні алгоритми для правильного функціонування гри та її розробки.

В даний час існує величезна кількість різних ігрових рушіїв. Основні їх відмінності полягають в підтримуваних мовах програмування, функціональності і, що не менш важливо, в вартості ліцензії. При виборі середовища розробки основна увага була приділена саме цим параметрам. Для аналізу обрані наступні ігрові рушії: Unreal Engine, Unity і Game Maker: Studio.

Game Maker: Studio - ігровий рушій, розроблений і підтримуваний компанією YoYo Games, написаний на Delphi (наступна версія на C#). На даний момент є одним з найбільш популярних ігрових рушіїв для розробки двовимірних ігор для всіх сучасних популярних платформ. Для розробки використовується власна спрощена мова програмування GML (Game Maker Language).

Даний ігровий рушій має кілька версій поширення:

- Trial;
- Desktop;
- Web;
- UWP;
- Mobile.

Кожна з цих версій (крім Trial) відрізняється платформою, для якої буде створена гра. Версія Trial є безкоштовною і надається для тих, хто хоче спочатку випробувати ігровий рушій, має ряд обмежень на кількість

використовуваних в грі об'єктів, а також дозволяє компілювати проект тільки для тесту на операційній системі Windows.

Unreal Engine - ігровий рушій, який розробляється і підтримується компанією Epic Games.

Перша гра, створена на цьому рушії, “Unreal”, з'явилася в 1998 році. З тих пір різні версії рушія були використані при розробці більш ніж сотні ігор та інших проектів.

Написаний на мові C++, рушій дозволяє створювати відеоігри для більшості операційних систем і платформ: Microsoft Windows, Linux, Mac OS і Mac OS X; консолей Xbox, Xbox 360, Xbox One, PlayStation 2, PlayStation 3, PlayStation 4, PSP, PS Vita, Wii, Dreamcast, GameCube та ін., а також для пристроїв, керованих системою iOS і Android.

Для спрощення портування рушій використовує модульну систему залежних компонентів та підтримує різні системи рендеринга (OpenGL, Direct3D, Pixomatic), відтворення звуку (DirectSound3D, EAX, OpenAL), засоби голосового відтворення тексту, розпізнавання мови, модулі для роботи з мережею та підтримуваними пристроями введення.

Він є безкоштовним, але якщо щоквартальний прибуток компанії розробника перевищує 3000 доларів, то необхідно передавати компанії Epic Games (розробники ігрового рушія Unreal Engine) п’ять відсотків від прибутку з продажів гри.

Unity - засіб для розробки двомірних і тривимірних ігор, що є однією з найбільш популярних на сьогоднішній день системою розробки. Дозволяє створювати додатки, що працюють під більшістю сучасних операційних системам (Windows, OS X, Windows Phone, Android, Apple iOS, Linux), а також на ігрових приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One і MotionParallax3D дисплеях (пристрої для відтворення віртуальних голограм), наприклад, Nettlebox. Є можливість створювати додатки для запуску в браузерах за допомогою спеціального модуля Unity (Unity Web

Player), а також за допомогою реалізації технології WebGL. Останні версії Unity дозволяють створювати додатки для окулярів віртуальної реальності.

Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Рушій підтримує дві сценарні мови: C# та JavaScript (модифікація). Розрахунки фізики здійснює фізичний рушій PhysX від NVIDIA.

Проект в Unity ділиться на сцени (рівні) - окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв і налаштувань. Сцени можуть містити в собі як, власне, об'єкти (моделі), так і порожні ігрові об'єкти - об'єкти, які не мають моделі («пустушки»). Об'єкти, в свою чергу містять набори компонентів, з якими взаємодіють скрипти.

Unity поширюється безкоштовно, але крім безкоштовної, існують чотири збірки - стандартна Unity, Unity iOS Pro, Android Pro і командна ліцензія. Вони відрізняються вартістю та функціональністю.

Безкоштовна версія має деякі обмеження, проте можливість поширювати ігри є, за умови, що щорічний дохід з гри не перевищує 100000 доларів. З виходом Unity 5 багато обмежень безкоштовної версії були прибрані, але щорічний дохід з гри все також не повинен перевищувати 100000 доларів.

Unity і Unreal Engine 4 на сьогоднішній день є найпопулярнішими ігровими рушіями. Це відмінні ігрові рушії, в залежності від того, яку гру потрібно розробити, кожен може стати для оптимальним варіантом.

Якщо потрібно розробити мобільну гру - Unity буде ідеальним рішенням. Це підтверджується домінуванням Unity серед розробників мобільних ігор, а також великою кількістю плагінів для використання нативних можливостей мобільних платформ: реклама, внутрішні покупки, аналітика, ігрові центри і т.д. - все це інтегрується в гру за лічені хвилини. Якщо хочеться розробити двовимірну гру, Unity теж буде прекрасним вибором, тому що саме у нього є прекрасні можливості для створення

двовимірних ігор. Хоча Unreal Engine 4 останнім часом намагається заманити розробників мобільних додатків, обіцяючи не менші, а навіть більші можливості для двовимірних ігор.

Якщо метою є створення красивої тривимірної гри, то краще обрати Unreal Engine 4, тому що цей рушій далеко попереду Unity, коли мова йде про розробку відеогри з так званою Next-Gen графікою. Але Unity теж потужний інструмент для розробки тривимірних ігор, хоча графічно він далеко не на тому рівні, що Unreal Engine 4.

У Unreal Engine 4 вбудована постобробка. До сцени можна застосовувати bloom-ефект, тонування і згладжування як глобально, так і до окремих її частин (за допомогою компонента PostProcessVolume). У Unity є стек постобробки, який можна завантажити з магазину ресурсів рушія. Система менш гнучка, ніж в Unreal Engine 4 - ефекти застосовуються тільки стеком або скриптами до камери.

В Unreal Engine 4 використовується мова програмування C++. У Unity в основному C#. Яка програма краще з точки зору мов програмування - дійсно зводиться до особистих вподобань. Якщо віддається перевага якійсь із цих мов, то вибір рушія може бути досить очевидним.

Але Unreal Engine 4 має рішення для людей, які бояться високого порогу входження в C++. Це Blueprint - редактор візуального скриптування. Технічно розробнику не потрібно писати жодного рядка коду. Це дуже зручно для створення швидких прототипів. Але за допомогою Blueprint можна створювати і повноцінні відеоігри, оскільки це дуже потужна і опрацьована система візуального скриптування. У Unity немає системи візуального скриптування, щоб використовувати щось подібне, розробник змушений купувати сторонні додатки на зразок Playmaker.

На відміну від Unity, в Unreal Engine 4 є ще один великий плюс для програмістів - це відкритий код.

Для розробки проекту був обраний ігровий рушій Unreal Engine 4, так як планується розробка тривимірної комп’ютерної гри для платформи PC

(персональний комп’ютер) і саме він надає величезний функціонал в створенні різного роду тривимірних відеоігор. До того ж наявність візуального середовища розробки скриптів (Blueprints) сильно допоможе при першому досвіді розробки комп'ютерної гри.

1.4. Аналіз існуючих розробок

Рушій Unreal Engine бере своє коріння ще в далекому 1998 року. З моменту його створення вийшло більше трьохсот комп’ютерних ігор, які розроблені на основі різних версій даного рушія. В якості проекту була обрана комп’ютерна гра на базі рушія четвертої версії Unreal Engine, тому основних конкурентів варто шукати серед відеоігор, які розроблені саме на базі Unreal Engine 4. Не дивлячись на те, що ця версія рушія вийшла в 2014 році, на його основі розроблено вже близько сотні ігор. Кращими комп'ютерними іграми, які розроблені на базі Unreal Engine 4, можна вважати:

- Dead by Daylight;
- PlayerUnknown's Battlegrounds;
- Sea of Thieves.

Dead by Daylight – комп’ютерна гра виключно з багатокористувацким режимом гри. Завдання у сторін прості. Четвірка людей повинна уникнути палких обіймів маніяка, запустити п'ять генераторів і скоріше втекти через один з відкрившихся виходів. Маніякові ж треба вистежити бідолах і акуратно розвісити по крюкам.

Ця гра жонглює умовностями і штампами фільмів-слешерів, вибудовуючи на них весь ігровий процес. Класична ситуація - герой щосили тікає від маніяка, той похмуро крокує слідом, але все одно наздоганяє жертву. В Dead by Daylight подібна хвилююча сцена є майже в кожному матчі. Невбиваємий (буквально) маніяк завжди трохи швидше жертв, він вміє читати залишені ними сліди, краще бачить карту і має куди більше часу для її

вивчення. Біжи не біжи, а шанс, що вбивця раптово з'явиться з-за рогу або схопить за п'яти, коли ти в розпачі лізеш у вікно, завжди великий.

Соковито встромивши мачете в плоть жертви, маніяк обов'язково зупиниться і витре інструмент об рукав, даючи пораненому герою невелику фору. До того ж одного удару завжди мало. Вбивця дуже повільно протискується в віконні отвори і довго ламає перешкоди, щоб жертва отримала ще один шанс і гра в кішки-мишки розтягнулася.

PlayerUnknown's Battlegrounds – це багатокористувацький шутер, з продуманою балістикою в дусі серії комп’ютерних ігор “ARMA”, з масою спорядження і зброї, що вимагає вивчення та акуратного використання, з мільйоном можливих тактичних схем. Тактику і хитрості “PlayerUnknown's Battlegrounds” можна розбирати нескінченно довго. Ця гра дуже швидко стала популярною, тому що це максимально проста в освоєнні розвага. Хаотичний симулятор виживання, в якому можна вмирати і вбивати тисячею дурних способів, а потім переказувати це друзям.

У чому суть “PlayerUnknown's Battlegrounds”? Ця гра відноситься до жанру королівська битва. Сотня гравців десантується на величезній карті: з полями, лісами, містами, селами, одиночними будинками і мостами. Маршрут у літака, на якому прибувають гравці, кожен раз різний, а момент для стрибка можна вибрати самому. Спорядження немає: броня, зброя, аптечки, обважування на зброю - все потрібно шукати на рівні, обшукуючи будинки. Згодом на карті виділяється область у вигляді кола, яке поступово звужується. Всі гравці, що знаходяться за його межами, досить швидко вмирають. Мета – стати останнім, хто виживе.

Це гра, яка створює історії. Гра про те, як ви з друзями перекинулися на позашляховику в пустелі, а на шум приїхали ще дві «банди» і вам довелося відстрілюватися, ховаючись за перевернутої машиною. Гра про те, як ти сам, взагалі без зброї, став єдиним, хто вижив, просто передавивши всіх на баггі. Гра про те, як ви разом з іншим гравцем висадилися в одній точці, де зброї не виявилося взагалі і довелося влаштовувати кулачний бій.

Родзинка “королівської битви” - саме зменшуваний згодом ігровий простір. Безпечне коло з кожним разом стає все менше, а за його межами гравці з кожним разом отримують все більший перманентні поранення. До того ж нова зона з'являється в межах існуючої в випадковому місці, а тому відсиджуватися всю гру в більшості випадків не вийде.

Така варіативність в динаміці геймплея не дає занудьгувати. А величезна карта та випадкова висадка супротивників кожен раз генерують нові ігрові ситуації. Розважайся сам і розваж іншого - головний принцип “PlayerUnknown's Battlegrounds”. Можливо, тому в неї куди приємніше грати компанією. У команді з чотирьох чоловік тебе не вб'ють з першого разу: завжди є шанс, що напарники встигнуть вилікувати тебе.

Гра дуже хороша. Начебто розмірене і неспішна, вона постійно тримає гравця в напрузі і вимагає повної зосередженості. Якщо ви нікого не зустріли в десяти будинках підряд, не розслабляйтеся, ціна неуважності - життя. Тут немає єдиної вірною тактики. Кожен матч не схожий на попередній і вам доведеться постійно підлаштовуватися під обставини.

Sea of Thieves – багатокористувацька гра з відкритим світом, в якій можна спробувати себе в ролі пірата. В цю відеогру хочеться закохатися з першого погляду - перші години тут найсильніші. Гра приголомшливо виглядає: художники та аніматори Rare проробили колосальну роботу, створивши шалено красивий і чарівний світ. Яскраві, соковиті кольори, чудове освітлення, зі смаком зроблені модельки NPC (неігрових персонажів) і піратів, - образ Sea of Thieves надовго закарбується у вас в пам'яті.

Головною особливістю гри є морські подорожі. Ви можете вибрати корабель, взяти на борт друзів, або не робити цього, і відправитися в шлях. Гра, звичайно, дає вам певні завдання, щоб направити гравця в потрібне русло, але це лише рекомендаційна форма. Ніхто вас не обмежує в подорожах, ніхто не вимагає діяти так, як хоче розробник. Ні, ви цілком можете самостійно брати і подорожувати по світу, заглядати на далекі острови, шукати тайники і все в такому дусі. Морські подорожі вкрай цікаві,

вони дають можливість розслабитися і вдосталь зануритися в атмосферу захопливих пригод. Звичайно, якщо у вас є друзі та великий корабель, то будь-яка подорож перетворюється на справжню бурю сміху та емоцій. Адже рано чи пізно хтось точно кине якір просто тому, що йому так захотілося.

Ігровий процес максимально простий і захоплюючий - ви можете брати завдання у різних фракцій і подорожувати в пошуках скарбів або всіляких цінностей, щоб заробити золото. Його потім можна буде витрачати на розмальовку корабля, наприклад. Завдання максимально цікаві і нагадують фільми про піратів - вам дають карту з хрестиком, а потім потрібно знайти це місце, відкопати скриню з золотом і ще врятуватися від місцевих скелетів, які захищають золото від піратів. Завдання дуже цікаві та динамічні, нічого подібного в інших іграх точно не знайти.

Спочатку всі думали, що “Sea of Thieves” заточена під кооператив і без вірних товаришів грати не вийде. Однак, боротися з морем і ворогами можна і без них - є компактний кораблик, яким керувати поодинці навіть простіше, його предостатньо, щоб проходити завдання і заробляти дублони. Звичайно, в компанії друзів набагато веселіше, адже смішні ситуації генеруються випадковим чином - то хтось випаде за борт, то хтось вирішить покерувати судном, поки капітан відійшов, то ще щось. Але, це абсолютно не обов'язковий елемент і якщо у вас немає друзів, охочих пограти, то можна сміливо заходити самостійно. Доречі, командне управління великим галеоном реалізовано відмінно: один рулить, другий керує вітрилами, третій дивиться на карту, останній працює на підхваті. Відчуваєш себе частиною машини, яка живе тільки за рахунок зусиль всіх своїх гвинтиків. Іноді мурашки на шкірі здатні викликати морські бої з іншими піратами (реальними гравцями), коли ти однією рукою стріляєш, а іншою латаєш пробоїни в днище свого судна.

Дуже радує той факт, що гра має єдині сервери для гравців з двох платформ: персональний комп’ютер з операційною системою Windows 10 та ігрова консоль Xbox One.

ВИСНОВОК ДО РОЗДІЛУ 1

Провівши аналіз існуючих розробок, стало зрозуміло, що рушій Unreal Engine 4 дозволяє розробляти безліч різноманітних, масштабних та унікальних комп’ютерних ігор. І ці ігри дійсно стають популярними, в них грають мільйони гравців з усього світу. Роблячи заключний висновок, можна зрозуміти, що для того що б гра була успішною, потрібно урізноманітнити продукт як геймплейно, так і по можливості додати цікавий і захоплюючий сюжет.

РОЗДІЛ 2

2. ОПИС ІГРОВОГО РУШІЯ UNREAL ENGINE 4

2.1. Основні факти

Unreal Engine, створений в 1998 році компанією Epic Games, є рушієм, створеним в першу чергу для жанру шутерів від першої особи. При цьому не існує обмежень і рушій може використовуватися для самих різних типів ігор.

Перша версія рушія була випущена разом з грою Unreal, створеної на ньому. З тих пір рушій пройшов через безліч змін і поліпшень, з його допомогою низкою компаній було випущено безліч ігор різноманітних жанрів і на різних платформах, а його найбільш пізня на сьогоднішній день версія - Unreal Engine 4 - активно підтримується компанією Epic Games.

Варто відзначити, що рушій є повністю безкоштовним для будь-яких некомерційних проектів (для платних відеоігор розробники рушія зобов'язують розробника гри віддавати 5% прибутку, але тільки в тому випадку, якщо дохід перевищує 3000 доларів в квартал). Unreal Engine 4 підтримує всі платформи та операційні системи, включаючи Windows, Linux, OS X, Android, Xbox One, PlayStation 4 і Ouya, а також підтримує безліч різних графічних API, такі як DirectX 11 і 12, OpenGL, Vulkan і JavaScript / WebGL. Unreal Engine дуже універсальний, і не дивлячись на те, що він трохи складніше у використанні, ніж Unity, він надає дуже великі функціональні можливості і неперевершену графіку.

В Unreal Engine 4 немає ніяких меж, ви можете створювати все, що ви захочете, будь то 3D-Puzzle або AAA-проект з відкритим світом. Unreal Engine 4 використовується студентами, інді-розробниками і великими командами, аудиторія цього рушія зростає з неймовірною швидкістю.

Unreal Engine 4 має повний набір інструментів розробки ігор, розроблений розробниками ігор для розробників ігор. Від двовимірних мобільних ігор до консольних блокбастерів і віртуальної реальності, Unreal

Engine 4 дає все необхідне для старту, просування і для того, щоб ваш проект був унікальним.

Одна з головних переваг рушія - це безліч інструментів і можливостей «з коробки». Не потрібно писати або купувати плагіни, досить завантажити редактор і вже можна приступати до розробки.

Unreal Engine 4 являє собою сукупність програм, редакторів, величезного набору бібліотек, докладної документації, безлічі уроків по розробці і використанню рушія, а також широкого спектру як платного, так і безкоштовного контенту, готового до інтеграції в призначених для користувача проектах [10].

Даний рушій має низку можливостей для розробки самих різноманітних ігор. Як і деякі інші рушії, Unreal Engine 4 включає в себе систему рендеринга, фізичний рушій, відтворення звуку, роботу з моделями, текстурами, анімацією, менеджмент ігрових ресурсів і управління пам'яттю, мережевий код та інші базові аспекти. Серед його особливостей можна перерахувати підтримку об'єктів, що руйнуються, оптимізовану і реалістичну обробку зіткнень між високим числом об'єктів складної форми, динамічне освітлення і безліч ефектів постобробки, редагування гри безпосередньо під час її виконання, систему штучного інтелекту, підтримку складних ландшафтів, систему матеріалів як сукупностей текстур і шейдерів, ієрархію об'єктів в ігровому просторі та багато іншого.

За допомогою Unreal Engine 4 можна як легко і швидко створювати робочі прототипи ігор, так і розробляти великі відеоігри, які не поступаються за якістю найсучаснішим стандартам. Рушій поширюється двома способами: у вигляді готового набору програм і у вигляді вихідного коду. У першому випадку користувачеві не потрібно самостійно проходити через досить тривалий процес компіляції всього рушія, однак при цьому втрачається можливість редагувати його безпосередній вихідний код, якщо це необхідно. У другому ж випадку користувач отримує доступ до повного вихідного коду рушія на мові C++, створивши копію вихідного коду в системі GitHub. Якщо

користувач додасть в рушій нову можливість або поліпшить одну з існуючих, то він також може запропонувати до включення в наступну версію рушія свої зміни.

2.2. Інтерфейс

Головний інтерфейс, який називається редактором рівнів (Level Editor), як правило, використовується для створення світу і рівнів, а також для розміщення ассетів (неподільних сутностей, які представляють частину ігрового контенту і мають якісь властивості). Головний інтерфейс редактора має сім ключових панелей: рядок меню (menu bar), панель Modes, панель World Outliner, панель Details, панель Content Browser, панель інструментів редактора рівнів і панель Viewport (див. рис. 2.1) [13].

Рисунок 2.1 – Level Editor

2.2.1. Рядок меню

Рядок меню, як і в більшості сучасних додатків, складається з пунктів File, Edit, Window і Help. File містить операції завантаження та збереження проектів і рівнів. Edit включає стандартні операції копіювання і вставки, а

також вподобання редактора та настройки проекту. Window відкриває вікно перегляду проекції (виюпорт) та інші панелі. Якщо ви закрили вікно або панель, то треба перейти в меню Window, щоб відкрити його знову. Help містить посилання на зовнішні ресурси, такі як онлайн-документація та посібники.

2.2.2. Панель Modes

Панель Modes відображає різні режими редактора рівнів. Вона дозволяє спеціалізованим інтерфейсам редагування працювати з відповідними типами акторів і геометрії (див. рис. 2.2).

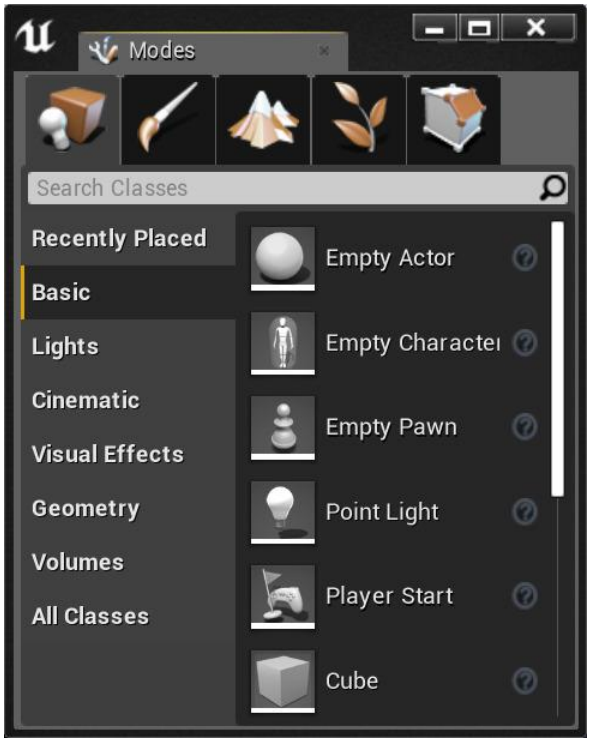


Рисунок 2.2 – Панель Modes

Панель Modes складається з різних інструментальних режимів редактора рівнів, які дозволяють змінювати його роботу. Тут ви можете вибрати спеціалізовану задачу, таку як розміщення нових ассетів в світ, скульптування ландшафтів, створення геометричних кистей і об'ємів, генерування рослинності або фарбування мешів (моделей). Всього редактор має п’ять режимів:

• Place – розміщення акторів на сцену;

• Paint – зміна кольору вершин в акторах статичних мешів;

• Landscape – редагування ландшафтних акторів;

• Foliage – додавання акторів рослинності на рівні;

• Geometry Editing – редагування акторів кисті по вершинах на рівні.

2.2.3. Панель World Outliner

Панель World Outliner відображає всіх акторів на поточному рівні в вигляді ієрархічного дерева (див. рис. 2.3). Ви можете вибрати актора, натиснувши по його назві на панелі World Outliner, його властивості відобразяться на панелі Details. Якщо двічі натиснути по назві, панель Viewport сфокусується на обраному вами ассеті.

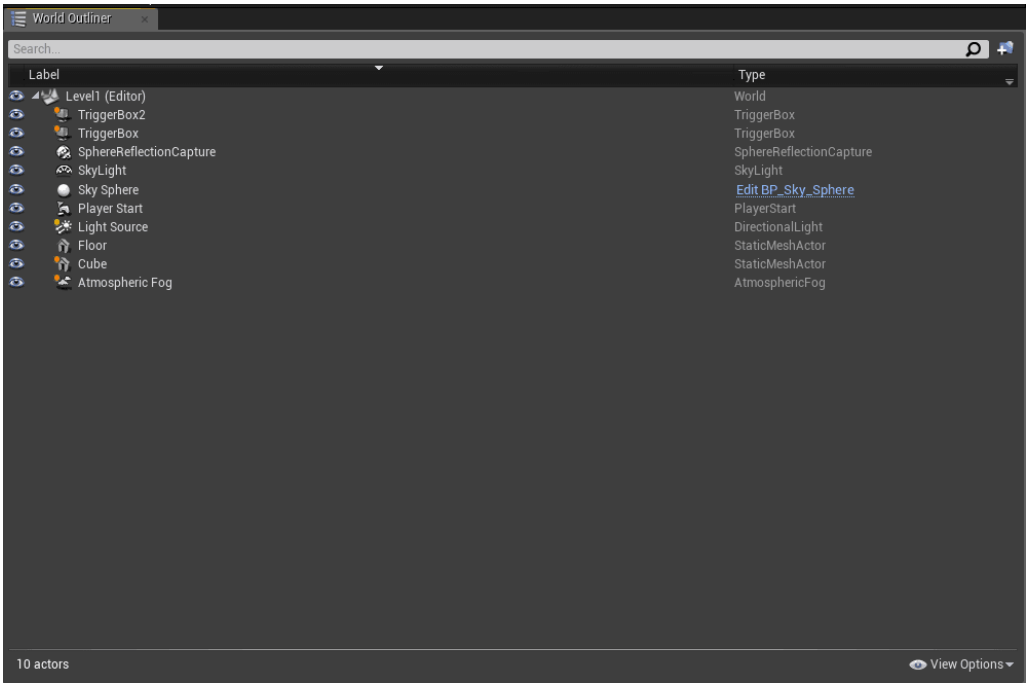


Рисунок 2.3 – Панель World Outliner

2.2.4. Панель Details

Панель Details є однією з найбільш часто використовуваних в Unreal Engine 4. Панель Details є практично в кожному підредакторі. Вона

відображає всі редаговані властивості обраних акторів на панелі Viewport. Ці властивості залежать від типу обраного актора, але існують деякі загальні властивості, наявні у більшості акторів (див. рис. 2.4). Типові властивості включають назву актора, блоки трансформації і редагування для переміщення, повороту і масштабування акторів, а також властивості відображення.

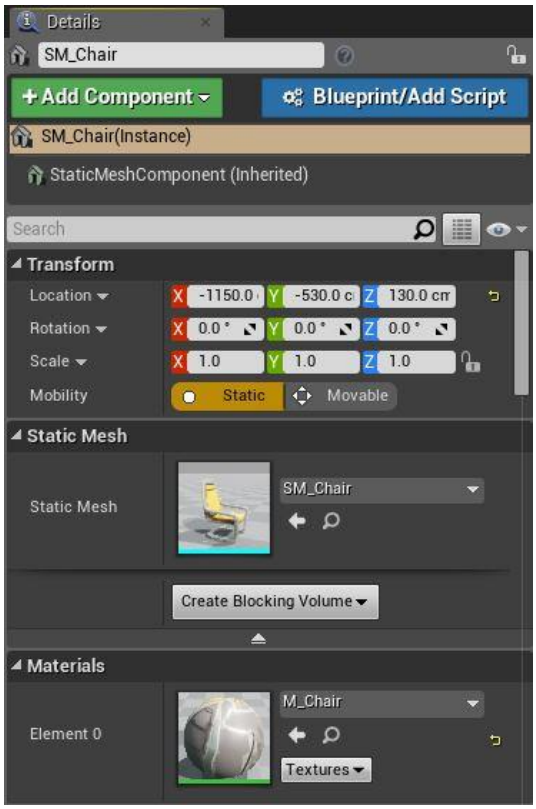


Рисунок 2.4 – Панель Details

2.2.5. Панель Content Browser

Панель Content Browser - основна область управління ассетами в проекті (див. рис. 2.5). Цей браузер використовується для задач, пов'язаних з контентом, таких як створення, перегляд, зміна, імпорт і організація. Він також дозволяє керувати папками і виконувати базові операції з ассетами, такі як перегляд посилань, переміщення, копіювання і перейменування. У Content Browser є рядок пошуку і прапорці фільтра для швидкого пошуку ассетів.

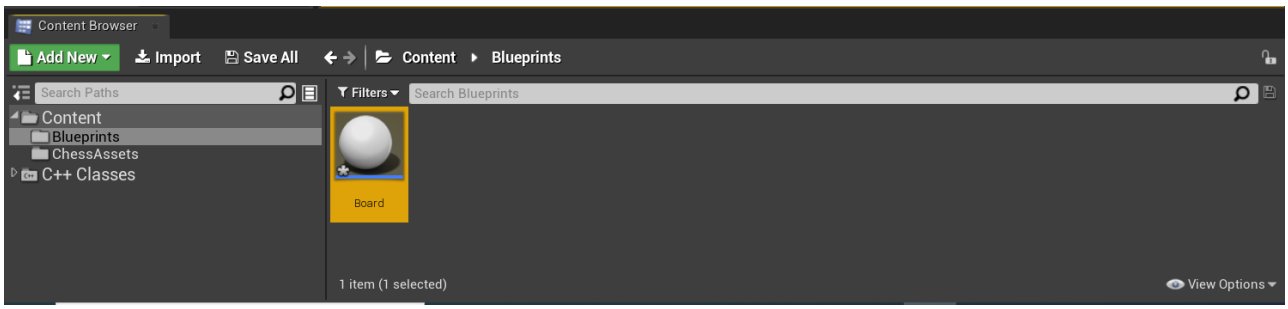


Рисунок 2.5 – Панель Content Browser

Можна уявити панель Content Browser як іграшкову коробку з нескінченними ассетами. При необхідності ви можете дістати з неї екземпляр (тобто копію) ассета і помістити його на рівень. Як тільки екземпляр буде поміщений на рівень, він стане актором. Початковий екземпляр розміщеного актора є точною копією оригінального ассету, що міститься на панелі Content Browser. У лівій частині панелі Content Browser знаходиться панель Source, яка відображає ієрархію папок контенту. Її можна розгорнути або згорнути, натиснувши по іконці в лівому верхньому кутку під зеленою кнопкою Add New. Права сторона панелі Content Browser називається Asset Management і показує ассети в обраній папці на панелі Source [8].

2.2.6. Панель Viewport

В’юпорти (Viewports) - це панелі для конструювання та перегляду створених вами світів. Панель Viewport використовується для переміщення по поточному рівню. Ця панель має безліч різних режимів, розташувань і налаштувань, які допомагають створювати і редагувати рівні, а також керувати ними (див. рис. 2.6) [14].

Рисунок 2.6 – Панель Viewport

2.3. Шаблони та демо-проекти

Відразу після запуску рушія запропоновується вибрати один із шаблонів. Можна, звичайно, відкрити порожній, але завжди легше переписувати і доповнювати щось готове, якщо шаблон підходить до вибраного жанру комп'ютерної гри. Всього пропонується дев'ять шаблонів:

- FirstPerson - шаблон під шутери від першої особи;
- Flying - шаблон під прості симулятори літаків;
- Puzzle - шаблон для логічної гри;
- Rolling - шаблон з м'ячем;
- Side Scroller - шаблон для платформера;
- 2D Side Scroller - шаблон для двовимірного платформера;
- Third Person - шаблон з видом від третьої особи;
- Top Down – шаблон з видом зверху і управлінням мишею;
- Twin Stick - шаблон з аркадою видом зверху;
- Vehicle - шаблон для створення авто-симуляторів;
- Vehicle Advanced - шаблон для створення авто-симуляторів з поліпшеною системою підвіски.

Також варті уваги демо-проекти з лаунчеру Epic Games (вкладка Learn / Вивчити). Там можна знайти дуже багато проектів на різні теми. Причому це практично повноцінні прототипи ігор, з логікою і графікою.

Демо ідеально підійдуть новачкам - їх можна вивчати або зробити свій проект на їх основі. Всі готові шаблони і демо можна використовувати в комерційних цілях.

2.4. Редактор

У перший раз тут можна загубитися – в основному редакторі Unreal Engine 4 величезна кількість різних налаштувань, а у кожного ассета є ще й внутрішній редактор для налаштування контенту. Наприклад, в окремому редакторі статичних мешів можна згенерувати низкополігональні моделі, а в редакторі анімацій створити пози або змінити анімацію.

Може здатися, що редактор перевантажений функціоналом, але в цьому і плюс: все необхідне знаходиться в одному місці. Хоча створювати дійсно хороший контент (моделі, звуки, текстури і т.д.) все одно доведеться в сторонніх програмах.

Зате в Unreal Engine 4 є інструменти для прототипування, завдяки яким можна зібрати гру взагалі без використання інших програм. Наприклад, за допомогою Brush'ей можна змоделювати базовий світ і предмети. А вже потім поміняти їх на більш якісні або залишити як є, якщо цього дозволяє стилістика проекту [4].

2.5. Об’єкти

При створенні рівнів або сцен, об'єкти розміщуються на карті, переміщаються для створення середовища, і їх параметри і властивості змінюються для потрібного формального вигляду і поведінки. Нижче представлені типи об'єктів, які найчастіше використовуються та зустрічаються при розробці рівнів і сцен на Unreal Engine 4.

StaticMesh Actor (СтатикМеш) – це простий тип об'єкта, який відображає трьохмірну модель на сцені. Назва зовсім не означає, що об'єкт не може переміщатися - під «статичним» розуміється статична геометрія об'єкта. Даний об'єкт може переміщатися або змінюватися будь-якими способами в процесі. Ці об'єкти в основному використовуються як об'єкти навколишнього світу або декорації при створенні сцени.

Brush – об'єкт (так само відомий як BSP Браш), що відображає тривимірний і статичний примітив на сцені. Ці об'єкти можуть бути змінені в режимі Редагування Геометрії (Geometry Editing mode) в редакторі рівнів. Браши використовуються в основному для швидкого макетування середовища і чорного створення рівнів для перевірки ігрового процесу [11].

SkeletalMesh Actor – тип об'єктів, що відображає анімований об'єкт зі скелетом, чия геометрія може бути деформована - в основному через використання анімаційних послідовностей, створених і імпортованих з інших програм для створення 3д-анімації. Ці об'єкти часто використовуються для створення персонажів або інших рухомих істот, а так само для комплексної механіки і всього, що має змінювати форму або відображати інший комплексний рух. Так само може використовуватися в зв'язці з Matinee для створення анімацій.

Player Start - об'єкт, який визначає, в якому місці буде починати гравець і де спочатку буде з'являтися головний персонаж [9].

Trigger – об'єкти, що використовуються для створення подій, що відбуваються при взаємодії будь-яких об'єктів в рівні. Іншими словами, вони запускають події, що відповідають за певну дію в рівні. Всі тригери за замовчуванням однакові і розрізняються лише формою зони впливу тригера на об'єкти, які його активують - коробка, капсула, сфера [7].

Matinee – об'єкти, доступні для завдання анімації властивостей об'єктів в часі за допомогою анімаційного редактора Matinee, і для створення як динамічних моментів ігрового процесу, так і анімованих сцен, “вшитих” в рівень. Система заснована на використанні спеціалізованих анімаційних

доріжок, в які можна розмістити ключові кадри для задання значень певних властивостей об'єктів рівня. Редактор Matinee близький до нелінійних редакторів відео, що робить його зрозумілим для професіоналів в області відео.

Point Light – працює дуже подібно звичайної електричної лампочки, поширюючи світло у всіх напрямках. Однак, для економії ресурсів, процес спрощений до освітлення у всіх напрямках тільки з однієї точки простору.

Spot Light – поширює світло з однієї точки в формі конуса. Користувачеві доступний один з двох конусів для освітлення - конус з внутрішнім кутом і конус із зовнішнім кутом. В межах конуса внутрішнього кута світло досягає повної яскравості. Як тільки ви виходите за межі внутрішнього радіусу, ви потрапляєте в конус зовнішнього кута, освітлення слабшає, створюючи півтінь, або пом’якшення освітлення навколо світлової плями. Радіус освітлення розділяє конуси. Простіше кажучи, це працює як ліхтар.

Directional Light – симулює світло, розповсюджуване джерелом, який знаходиться дуже далеко. Це означає, що всі тіні, що відкидаються цим джерелом будуть паралельні, що ідеально при створенні сонячного світла. Може мати один з трьох видів рухливості при розміщенні.

Particle Emitter – об'єкт, який використовується для створення таких ефектів, як дим, вогонь, іскри та ін., шляхом створення частинок в формі спрайтів (обличчям до камери) або об'єктів. Така поведінка частинок визначається і контролюється спеціальною системою - Particle System, яка створюються в Content Browser і змінюється в редакторі Cascade.

Ambient Sound – використовуються для відтворення звуків в просторі. Ці звуки можуть бути зациклені чи ні, мати поширення в просторі і посилення / затухання. Все це повинно бути задано в звуковому сигналі і не є в самому звуковому об'єкті [3].

Decal – спеціальний об'єкт, який проектує матеріал на іншу поверхню без необхідності використання геометрії. Її можна використовувати,

наприклад, для нанесення різних патьоків, тріщин, вм'ятин або бруду на геометрію, що б зробити поверхню більш реалістичною і з малою кількістю повторів.

Camera Actor – використовується для того, щоб відображати сцену на екран. Вона може існувати окремо і використовуватися разом з інструментом Matinee, а так само може існувати в якості компонента всередині інших об'єктів, наприклад всередині персонажа.

Volume – спеціальна зона, в якій діють будь-які постійні правила. Ці правила можуть бути як і для ігрового процесу (наприклад нанесення шкоди персонажу або зміна його типу переміщення), так і для редактора (позначати, де прораховується світло або навігація для штучного інтелекту).

Target Point – використовується для позначення будь-якої точки в просторі для подальшого використання в коді або Блупринтах. Наприклад, дизайнер рівня може поставити таку точку, що саме там відновлюється персонаж після смерті, програміст в свою чергу використовує цей об'єкт для визначення координат відновлення.

2.6. C++ чи Blueprints

При розробці комп'ютерної гри основним є написання її логіки. Зазвичай для цього використовується мова програмування, конкретно в Unreal Engine 4 ця мова - C++. Але, в Unreal Engine 4 ще є Blueprints – спеціальний інструмент візуального програмування. Він дозволяє будувати логіку гри за допомогою блок-схем з нод (вузлів).

З блупринтами працювати набагато легше, тому що в них неможливо допустити синтаксичну помилку - подати неправильний тип даних або забути поставити потрібний знак. Ще блупринти захищають від “вильотів” програми. Наприклад, якщо в C++ спробувати отримати доступ до неіснуючого об'єкту, гра вилетить, а в блупринтах просто з'явиться помилка в логах.

Але незважаючи на низький поріг входу і наочність, блупринти - повноцінний інструмент об'єктно-орієнтованого програмування (ООП).

Воно вимагає практики, щоб правильно все організовувати, розділяти по класах, компонентам і тримати в чистоті. Крім того, за допомогою блупринтів не вийде відправити зовнішній запит (наприклад, по http), попрацювати з файлової системою комп'ютера або підключити сторонню бібліотеку. Для цього доведеться качати плагіни або писати їх самостійно. Але в іншому можна обійтися і стандартними блупринтами - всередині рушія вони вміють все, що потрібно.

Розробка логіки гри на C++ займе дуже багато часу. Оскільки існує ймовірність наштовхнутися на велику кількість проблем, які буде дуже важко вирішити при першому досвіді розробки ігор і при тому, що документації по C++ у контексті Unreal Engine 4 досить мало (чого не скажеш про документацію по Blueprints). Тому було вирішено при розробці логіки гри використовувати саме Blueprints. Гру спокійно можна написати тільки на блупринтах (причому не тільки одиночну, а й мережеву) і в багатьох випадках різниці в продуктивності не буде.

2.7. Blueprints

Блупринти є типом ассетов, які надають користувачам інтуїтивну систему для створення спеціальних типів об'єктів, які внаслідок можна поставити на сцену. Так само блупринти використовуються для створення ігрової логіки рівня або логіки рівня.

Блупринти використовують вбудовану в редактор систему для візуальної побудови логічних послідовностей. З’єднуючи блоки, події, функції і змінні, можливо створити досить складні логічні елементи, які будуть створювати геймплей гри [1].

Два найбільш використовуваних типи блупринтів:

- Level Blueprint;
- Class Blueprint.

Кожен рівень має свій блупринт рівня. Сам Level Blueprint використовується для маніпулювання об'єктами на сцені в процесі гри, так само для контролю Matinee, стрімінг рівнів, чекпоінтів та інших систем, які відносяться до рівнів. Level Blueprint так само можуть взаємодіяти з Class Blueprint, які є на сцені.

Class Blueprint дозволяють створити складні об'єкти для подальшого розміщення на сцені, такі як двері, що відкриваються, ящики з предметами, кнопки і т.п. Наприклад, кнопка на підлозі та двері є різними блупринтами та містять певний скрипт для взаємодії з гравцем, програвання анімації і звуку, відкриття дверей і так далі.

В даному випадку, натискання кнопки активує подію усередині блупринта дверей, відкриваючи їх. З тим же успіхом можна зробити взаємодію будь-яких блупринтів, а так само спрацювання подій не тільки на гравця, але і на інші об'єкти або на Level Blueprint. Class Blueprint можуть бути повністю індивідуальними, а значить, для їх роботи не обов'язково впливати ззовні і вони можуть працювати і виконувати будь-які дії самі по собі.

Ігрові персонажі теж є Класовими Блупринтами, за допомогою яких можна створювати всі потрібні елементи і логіку майбутнього персонажа. Можна встановлювати параметри камери, встановлювати управління персонажем, включаючи мишу чи навіть сенсорні екрани і створювати речі, на які персонаж здатний.

При створенні Блупринта Персонажа (Character Blueprint) у вас будуть вже заготовлені властивості, що налаштовуються, для пересування, стрибка, плавання і падіння. Все, що потрібно, це додати управління і визначити, як ваш персонаж буде вести себе.

Інтерфейс створюється так само за допомогою блупринта, схожого на класовий, проте він прив'язується до геймплею, що не вимагає розміщення його на рівні. Можна встановити блупринт інтерфейсу так, щоб він читав змінні з інших блупринтів і відображав за допомогою цієї інформації будь-

які речі, наприклад бали, показник життів персонажа, кількість наявних патронів в запасі і так далі. За допомогою HUD (інтерфейс) блупринта можна так само створювати кнопки, щоб створювати різного роду меню [2].

Створюючи будь-яку логіку за допомогою Blueprints, ви будете працювати з Редактором Блупринтів (див. рис. 2.7). Редактори бувають різних типів, в залежності від редагованого блупринта. Основний функціонал редактора залишається скрізь (Графіки, змінні і т.д.), однак деякі блупринти, особливо Level Blueprint не мають своїх властивостей або компонентів.

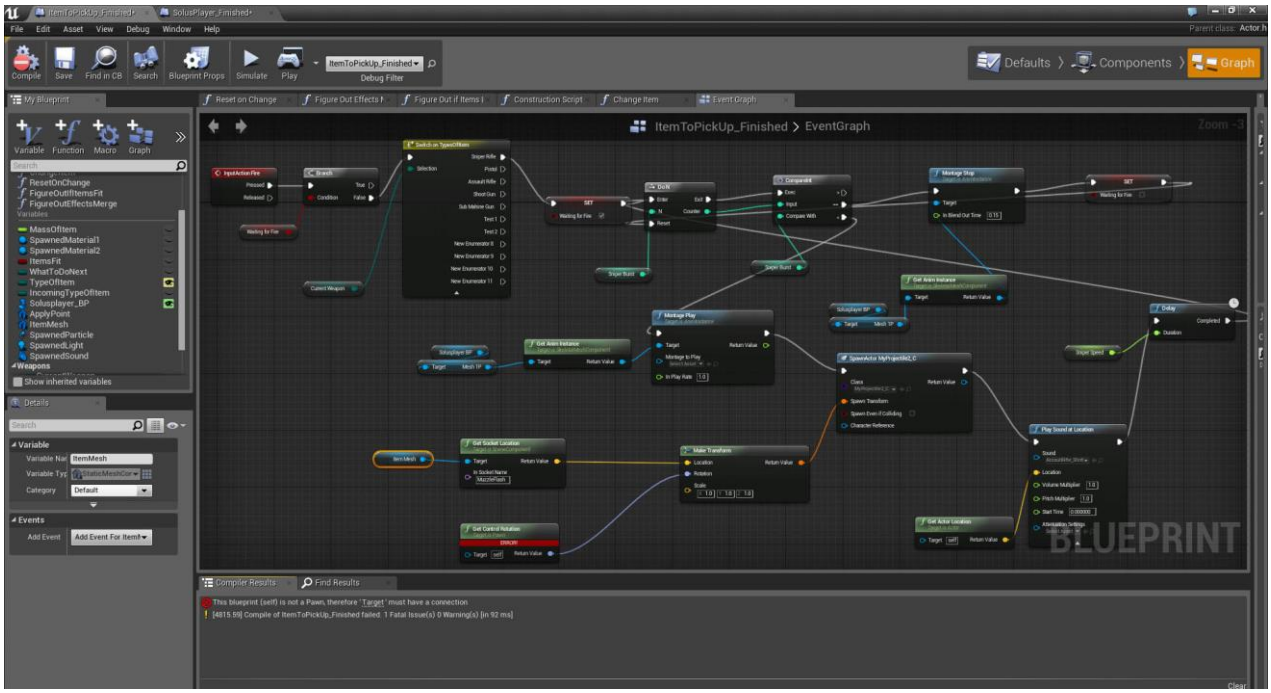


Рисунок 2.7 – Редактор Blueprints

2.8. Меши, матеріали та ефекти

Припустимо, що є вже готовий прототип ігрового світу і механіка геймплея. Тепер потрібно імпортувати моделі і налаштувати матеріали до них, щоб все виглядало як треба. Можна обійтися готовими ассетами.

Рушій підтримує .fbx і .obj формати 3D-моделей, а також всі популярні формати текстур. Імпортувати їх досить просто - можна навіть закинути все в папку з проектом і імпорт відбудеться автоматично.

Ще важливо налаштувати матеріали, тобто шейдери. Зазвичай вони пишуться кодом в інших програмах, але в Unreal Engine 4 і на цей випадок є візуальний інструментарій, який дозволяє описати інструкції шейдера нодами.

Не забудемо і про візуальні ефекти. Зараз в Unreal Engine 4 для них є два редактора: модульний і нодовий. У першому збирається система частинок і ефектів за допомогою модулів і налаштувань. А в другому можна створювати ефекти за допомогою нодів редактора, дуже схожого на редактор матеріалів. Там можна гнучко налаштовувати ефекти для отримання потрібного результату.

2.9. Інтерфейс користувача (UI)

В Unreal Engine 4 є спеціальний редактор віджетів - UMG. Як і все інше, він простий і потужний. Можна розміщувати елементи з якорями, можна створювати блочну верстку, вбудовувати віджети всередині інших віджетів і так далі. Є безліч різних налаштувань, майже все можна переробити або анімувати.

За допомогою UMG можна створити меню, HUD і інші частини призначеного для користувача інтерфейсу, а налаштувати роботу всього цього можна через ті ж блупринти.

2.10. Звук

Значну частину атмосфери гри створюють звукові ефекти. Зі звуком працювати теж не складно, достатньо імпортувати його форматом Wav 16 і програвати за допомогою блупринтів (або коду на C++) в потрібному місці. Або поставити компонент звуку і вмикати / вимикати його в потрібні моменти. Міксувати і створювати комплексні звукові ефекти в Unreal Engine 4 теж можна - знову ж таки, за допомогою нодів редактора [5].

2.11. Збірка

Unreal Engine 4 дозволяє зібрати проект під всі популярні платформи, від ПК і консолей до мобільних пристроїв і навіть HTML5. Для ПК вистачає одного кліка, а ось для інших платформ може знадобитися додатковий SDK, інструкції по якому є в документації [12].

ВИСНОВОК ДО РОЗДІЛУ 2

В Unreal Engine 4 є ще багато корисних інструментів, були розібрані тільки найосновніші. В цілому, це один з найпривабливіших рушіїв на сьогоднішній день - багато корисних інструментів і легкий в освоєнні.

РОЗДІЛ 3

3. РОЗРОБКА КОМП'ЮТЕРНОЇ ГРИ У ЖАНРІ ШУТЕР

Unreal Engine 4 встановлюється та запускається в подальшому через Epic Games Launcher. Перед тим, як почати розробку шутера завантажуємо Animation Starter Pack з магазину в лаунчері (див. рис. 3.1). Це спеціальний пак, який додасть в гру різну анімацію, в тому числі анімацію ходьби, стрибків, стрільби і так далі. Він є абсолютно безкоштовним і підготовлений самою компанією, яка є розробником рушія Unreal Engine.

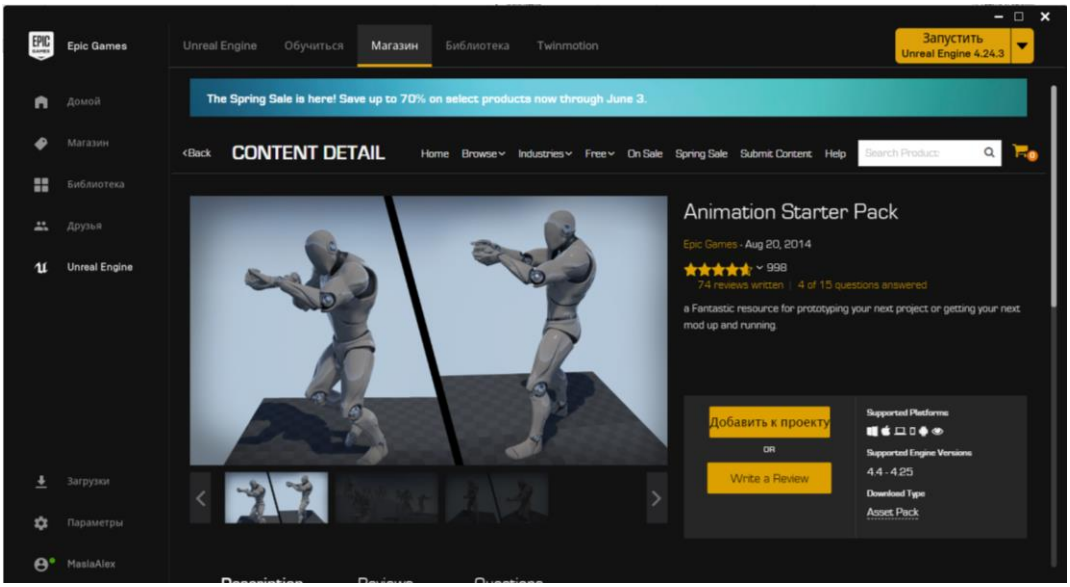


Рисунок 3.1 – Animation Starter Pack

3.1. Створення проекту

Спочатку потрібно обрати тип проекту. Оскільки ми будемо розробляти комп'ютерну гру, обираємо тип Games (див. рис. 3.2).

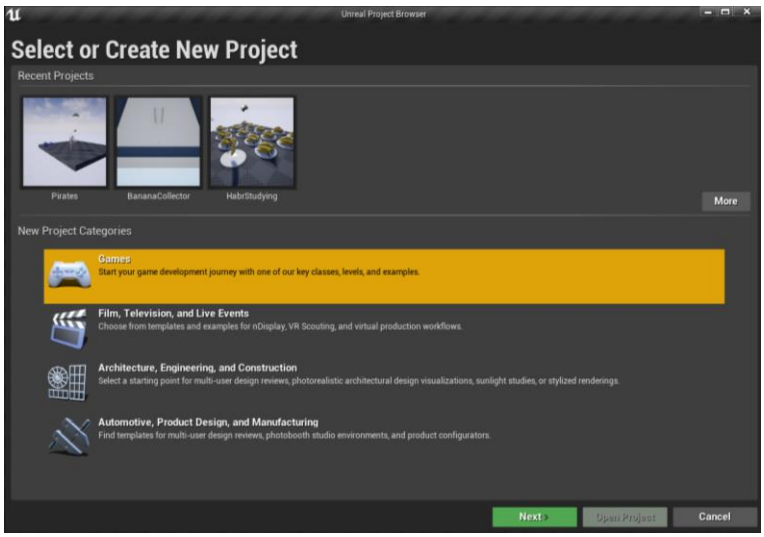


Рисунок 3.2 – Категорії проекту

У наступному вікні відразу ж пропонується на вибір кілька заготовлених шаблонів (див. рис. 3.3). Вибираємо шаблон Third Person. В такому випадку ми відразу отримаємо готового ігрового персонажа з видом від третьої особи.

Рисунок 3.3 – Вибір шаблону

В наступному вікні потрібно виставити основні налаштування проекту (див. рис. 3.4). Вказуємо, що при розробці будуть використовуватися Блупринти, а сам проект розроблятиметься для персональних комп’ютерів. Встановлюємо максимальну якість графіки, але трасування променів відключаємо. Також вказуємо, що нам потрібен Starter Content – стандартній

пакет, що містить прості розміщувані меші з основними матеріалами та текстурами. Далі вказуємо розміщення та назву проекту.

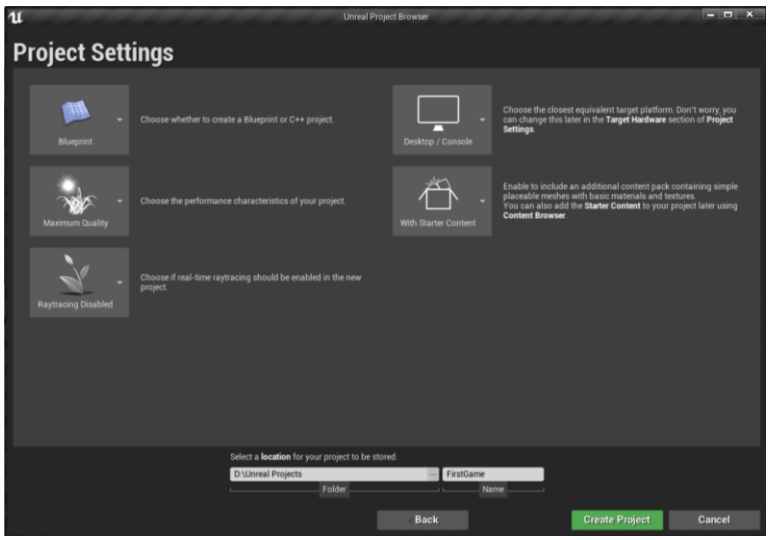


Рисунок 3.4 – Налаштування проекту

Після створення проекту відразу відкривається редактор створеного рівня, на якому можна побачити вже кілька елементів (акторів):

• Directional Light - направлене джерело світла, яке імітує світло, що йде здалеку;

• SphereReflectionCapture - фіксує, як виглядає довкілля з деякої точки. Потім це захоплення використовується як відображення на поверхнях, які знаходяться в радіусі впливу цього актора;

• AthmosphericFog - дає наближення розсіювання світла через планетарну атмосферу. Це може надати зовнішнім рівнями більш реалістичний вигляд;

• SkyLight - захоплює віддалені частини рівня і застосовує це до сцени в якості світла. Це означає, що зовнішній вигляд неба і його освітлення/відображення будуть відповідати, навіть якщо небо виходить з атмосфери, або будуть шаруваті хмари поверх хмарочоса або далеких гір;

• ThirdPersonCharacter - ігровий персонаж з налаштованими анімаціями, яким можна керувати;

• FollowCamera - представляє точку зору гравця, тобто як гравець бачить світ;

• StaticMeshComponent - використовується для створення екземплярів UStaticMesh. StaticMesh є частиною геометрії, яка складається з статичного набору багатокутників і є базовим елементом, використовуваним для створення геометрії світу для рівнів в Unreal Engine 4;

• LightmassImportanceVolume - керує областю, в яку Lightmass (створює світлові карти зі складними світловими взаємодіями, такими як затінення області і дифузна інтеррефлексія) випромінює фотони, дозволяючи сконцентрувати її лише на тій ділянці, яка потребує детального непрямого освітлення;

• PostProcessVolume - особливий тип обсягу (Вольюм). В Unreal Engine 4 кожен обсяг постобробки - всього лише один з типів шарів, які можуть створюватися за допомогою ігрового коду (ефект попадання), призначеного для користувача інтерфейсу (меню паузи), камери (віньєтка) або Матіні (matinee; ефект старовини). Кожен окремий шар має свою вагу, що дозволяє відмінно управляти ними, змішувати їх.

На даному етапі на рівні знаходиться простір з підлогою, який обмежений стінами та ігровий персонаж. Також присутні кілька предметів, на які гравець може стрибнути, бігати по ним і зістрибувати з них.

Якщо натиснути на кнопку Play, яка розташована на панелі інструментів (Toolbar), то прямо у вікні перегляду редактора активного рівня запуститься режим гри в нормальному стані. Після запуску одразу можна управляти ігровим персонажем, використовуючи клавіатуру для управління рухами і мишу для зміни положення камери.

Останнім кроком перед початком розробки додаємо до проекту анімаційний пакет Animation Starter Pack.

3.2. Налаштування камери

Так як вирішено розробити гру з видом від першої особи, але зараз в нашій грі вид від третьої особи. Треба це виправити. Для цього достатньо перемістити закріплену за персонажем камеру таким чином, щоб гравець бачив світ як би через очі персонажа, а не знаходячись як би позаду персонажу, як це зроблено у шаблоні.

Для цього на панелі Content Browser відкриваємо папку “Blueprints” (див. рис. 3.5), в якій знаходяться всі об’єкти з блупринтами, які є в нашій грі (див. рис. 3.6). Відкриваємо ThirdPersonCharacter і в новому вікні переходимо до вкладки Viewport.

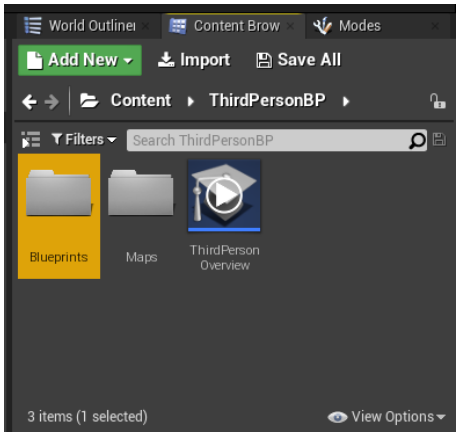


Рисунок 3.5 – Content Browser

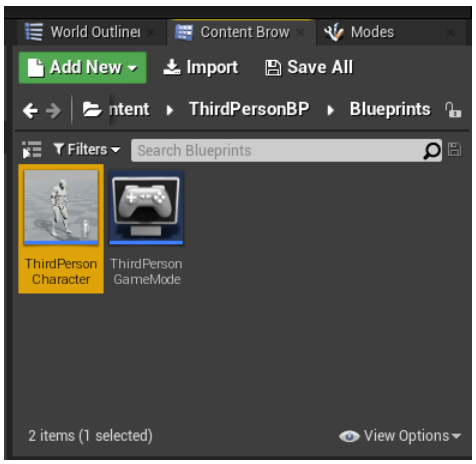


Рисунок 3.6 – Папка “Blueprints”

Спочатку переміщаємо камеру в потрібне положення, а потім робимо її дочірнім об’єктом персонажа (див. рис. 3.7). Оскільки камера статична, а персонаж має анімацію, треба зробити так, щоб камера рухалася разом з рухами голови персонажа. Для цього в параметрі камери Parents Socket обираємо “head”, щоб камера відстежувала анімаційні рухи голови персонажа та повторювала їх. Після цього камера автоматично перевертається та зміщується (див. рис. 3.8), тому треба повернути її на потрібне місце.

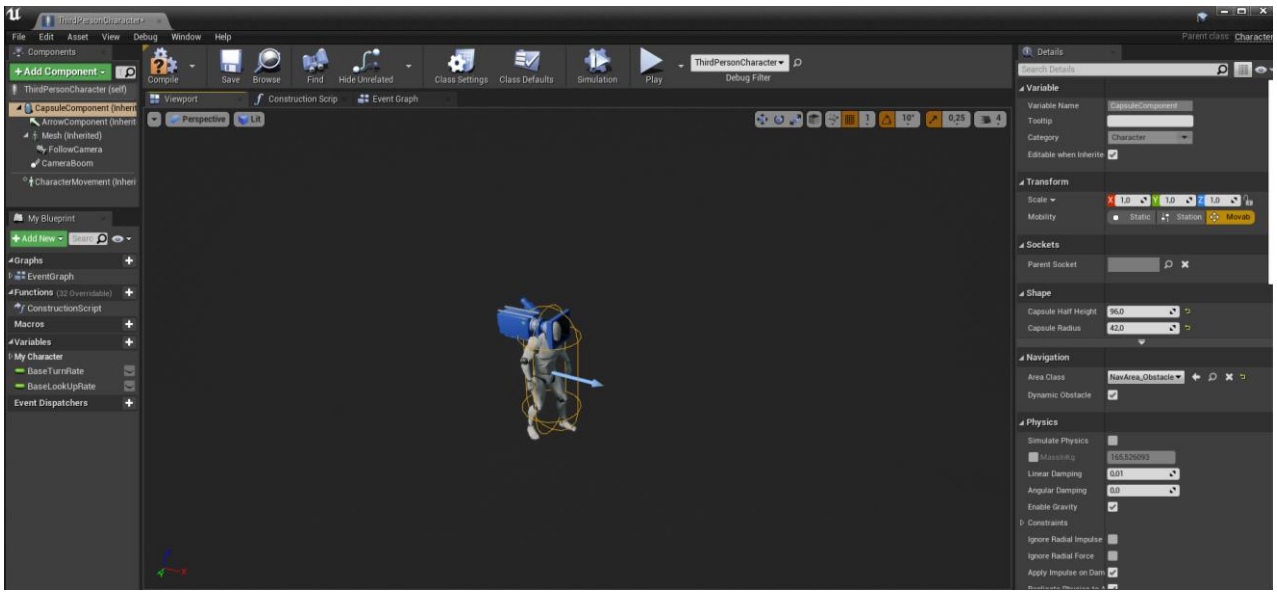


Рисунок 3.7 – Переміщення камери

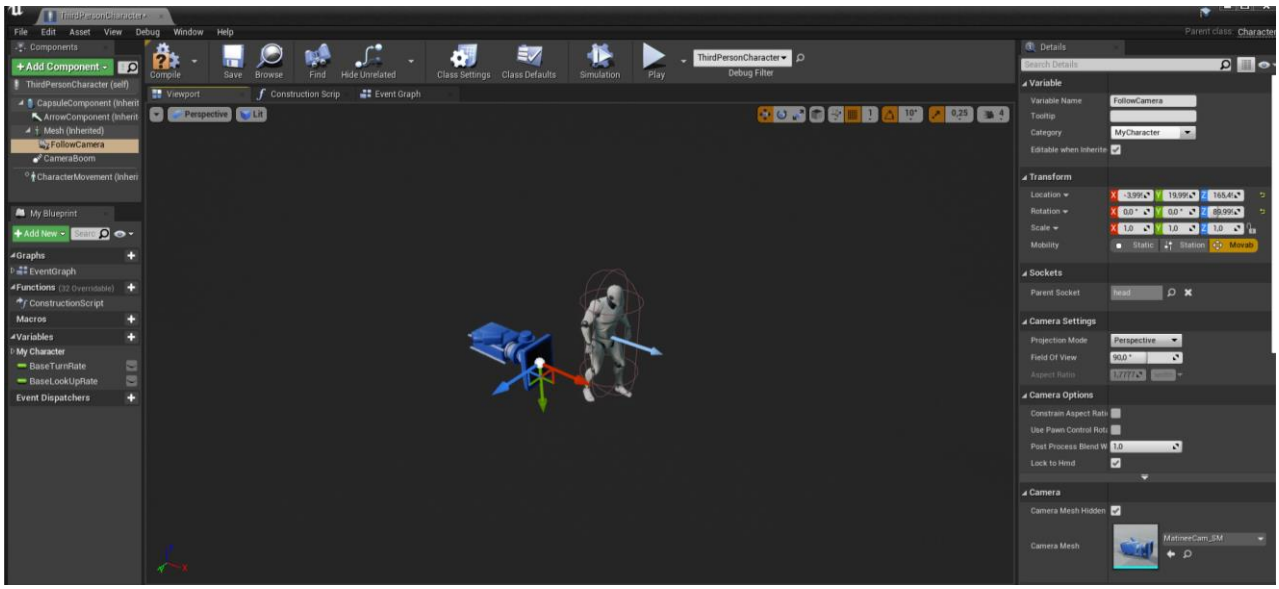


Рисунок 3.8 – Автоматичне змішення камери

За замовчуванням, компоненти Camera не використовують поворот контролера. Щоб виправити це, треба, обравши в редакторі камеру, на панелі Details активувати пункт “Use Pawn Control Rotation” (див. рис. 3.9). Далі в редакторі ThirdPersonCharacter на панелі Details потрібно зробити активним пункт “Use Rotation Controller Yaw”, щоб при поворотах камери тіло персонажа теж поверталося (див. рис. 3.10).

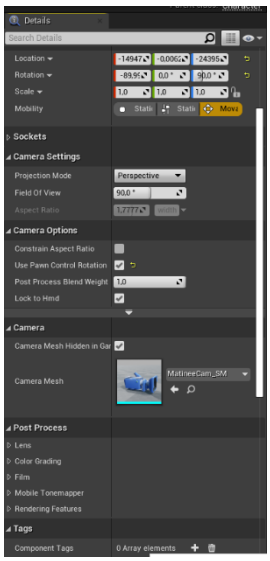


Рисунок 3.9 – Use Pawn Control Rotation

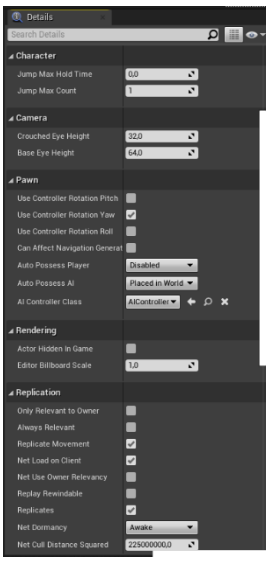


Рисунок 3.10 – Use Rotation Controller Yaw

Тестуємо, натиснувши кнопку Play на панелі інструментів. Можна побачити, що тепер в грі вид від першої особи (див. рис. 3.11).

Рисунок 3.11 – Вид від першої особи

3.3. Шкала життя і обладунків гравця

У грі буде дві шкали. Шкала життя і шкала обладунків. Обидві шкали будуть зменшуватися при нанесенні урону по гравцеві. Спочатку від одержуваного урону буде зменшуватися шкала обладунків, яка часом може

відновлюватися, але якщо вона повністю закінчиться, то буде зменшуватися шкала життя гравця, яка не має властивість відновлюватися. Якщо шкала життя дійде до нуля відсотків, то гравець помре.

Відкриваємо блупринт ThirdPersonCharacter (гравця) і в новому вікні переходимо до вкладки Event Graph. Спочатку треба додати дві змінні типу Float: Armor (обладунки) та Health (життя) (див. рис. 3.12).

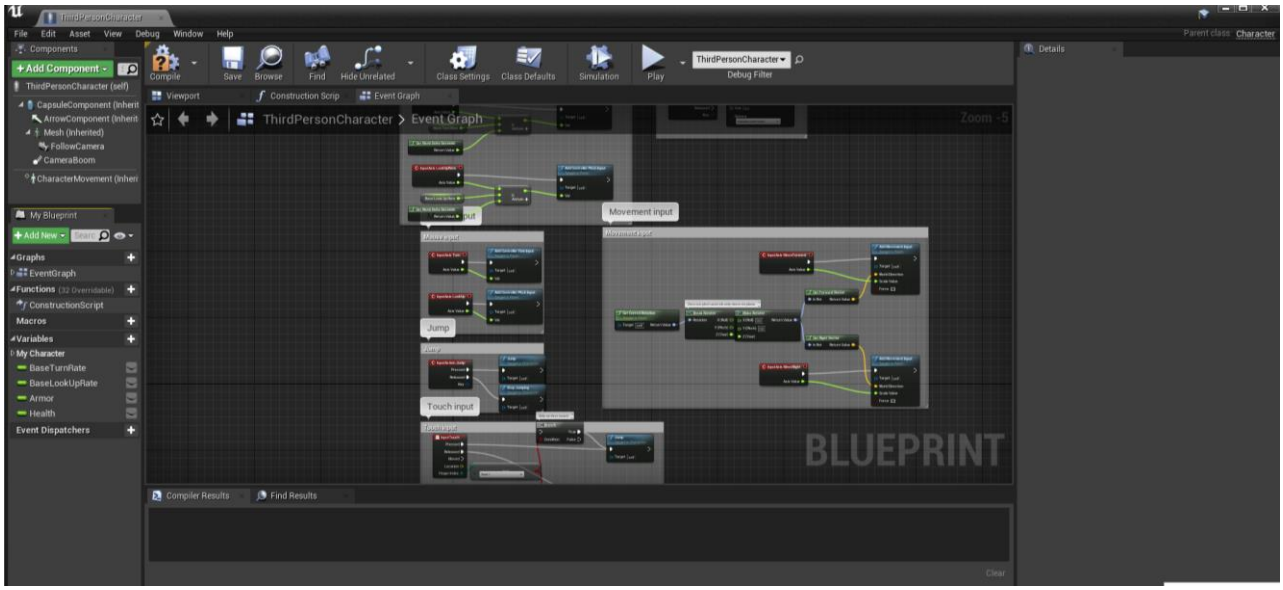


Рисунок 3.12 – Редактор ThirdPersonCharacter

Далі можна скомпілювати блупринт, натиснувши на кнопку Compile на панелі інструментів, та виставити початкові значення змінних. Нехай для обох змінних початковим буде значення 1.0. Тепер знову компілюємо та зберігаємо цей блупринт.

Для того, щоб створити інтерфейс гравця зі шкалами життя, створюємо клас типу Widget Blueprint, назвемо його HUD. В редакторі інтерфейса додаємо два елементи Progress Bar, налаштовуємо їх розмір, положення, колір. Це і будуть шкали, жовта – шкала обладунків, а червона – шкала життя (див. рис. 3.13).

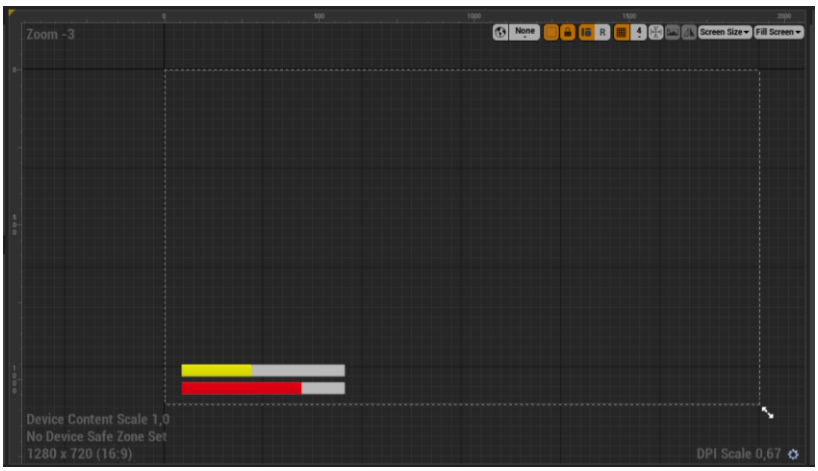


Рисунок 3.13 – HUD

Тепер треба прив’язати інтерфейс до змінних. Для цього використовується Blueprints – візуальне програмування. Було створено по одній спеціальній функції для кожної шкали, які звертаються до гравця, як до об’єкту, дізнаються значення потрібної змінної та повертають його (див. рис. 3.14, 3.15).

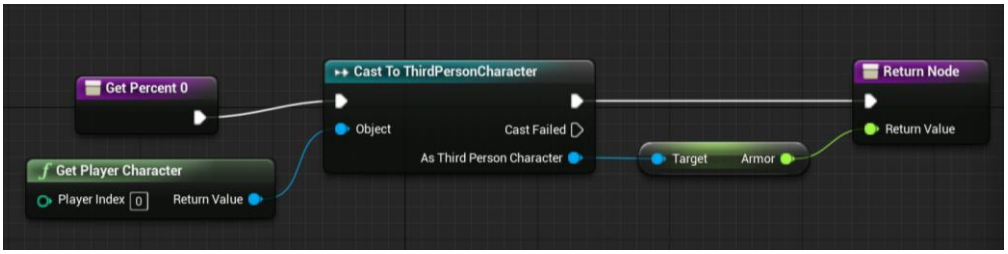


Рисунок 3.14 – Функція для шкали обладунків

Рисунок 3.15 – Функція для шкали життя

Тепер потрібно зробити так, щоб створені шкали відображались на екрані під час гри. Для цього доповнюємо блупринт персонажа (див. рис. 3.16).

Рисунок 3.16 – Скрипт для відображення шкал на екрані

Далі перевіряємо, що все працює (див. рис. 3.17).

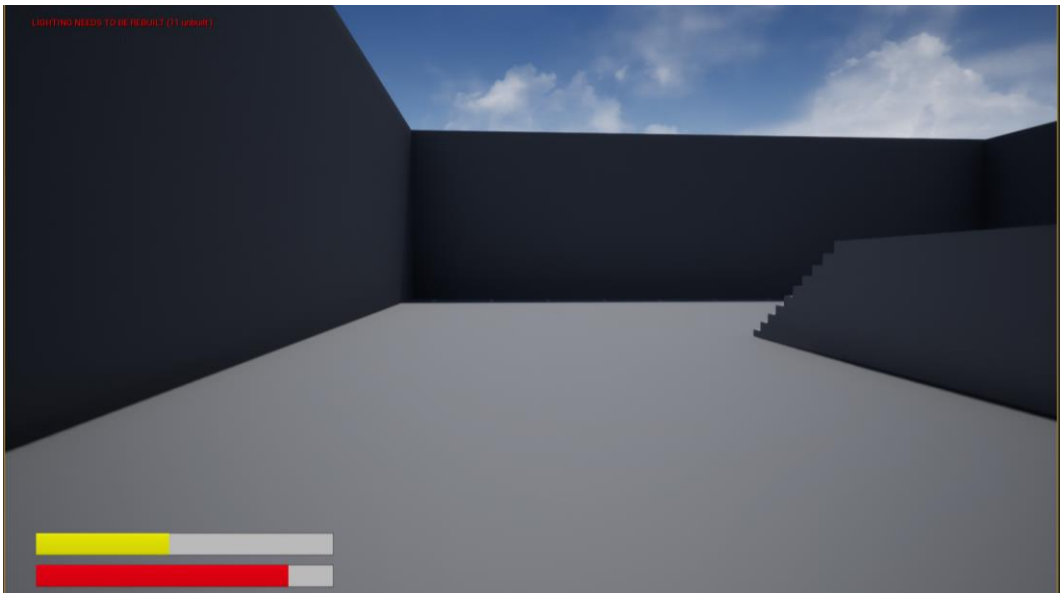


Рисунок 3.17 – Відображення шкал на екрані

3.4. Відновлення обладунків

Відновлення обладунків реалізується доповненням блупринта персонажа, а саме створенням графів (скриптів). Було вирішено зробити так, що якщо рівень обладунків менше 100%, то кожну секунду вони будуть відновлюватися на 1% (див. рис. 3.18).

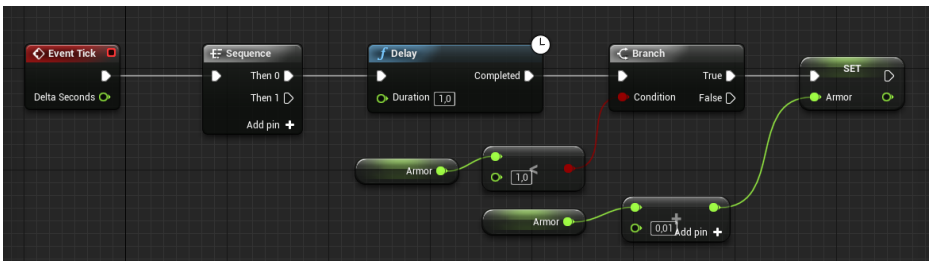


Рисунок 3.18 – Скрипт відновлення обладунків

3.5. Нанесення урону

Тепер для блупринта персонажа гравця треба створити функцію, яку можна буде визивати. Було вирішено розробити функцію, яка віднімає 5% від значення обладунків, а якщо обладунки мають нульове значення, то відніматися буде 5% від рівня життя персонажа (див. рис. 3.19).

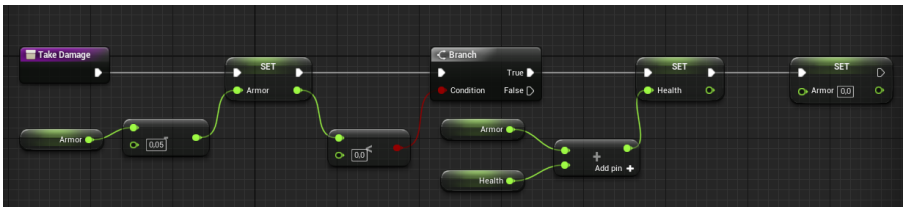


Рисунок 3.19 – Функція нанесення урону

3.6. Додавання повноцінного 3D персонажа з анімацією

Зараз наш персонаж має анімації бігу, стрибків та поворотів. Для простого шутера цього може бути достатньо, але спочатку потрібно змінити анімації таким чином, щоб вони виглядали реалістично, коли персонаж буде тримати в руках автомат. Це доволі довго та складно, треба бути професіоналом. Тому було вирішено завантажити готового персонажа з готовими анімаціями для шутера, який знаходиться в інтернеті у відкритому доступі. Але, це все потребує налаштування.

Спочатку створюємо ассет Animation Blueprint. Відкривши цей блупринт, бачимо редактор (див. рис. 3.20), в якому можна налаштовувати, які анімації та коли будуть активними. В один момент часу програватися

може лише одна анімація, але можна створити State Machine (скінченний автомат), щоб налаштувати переходи від однієї анімації до іншої, в залежності від ігрових ситуацій [15].

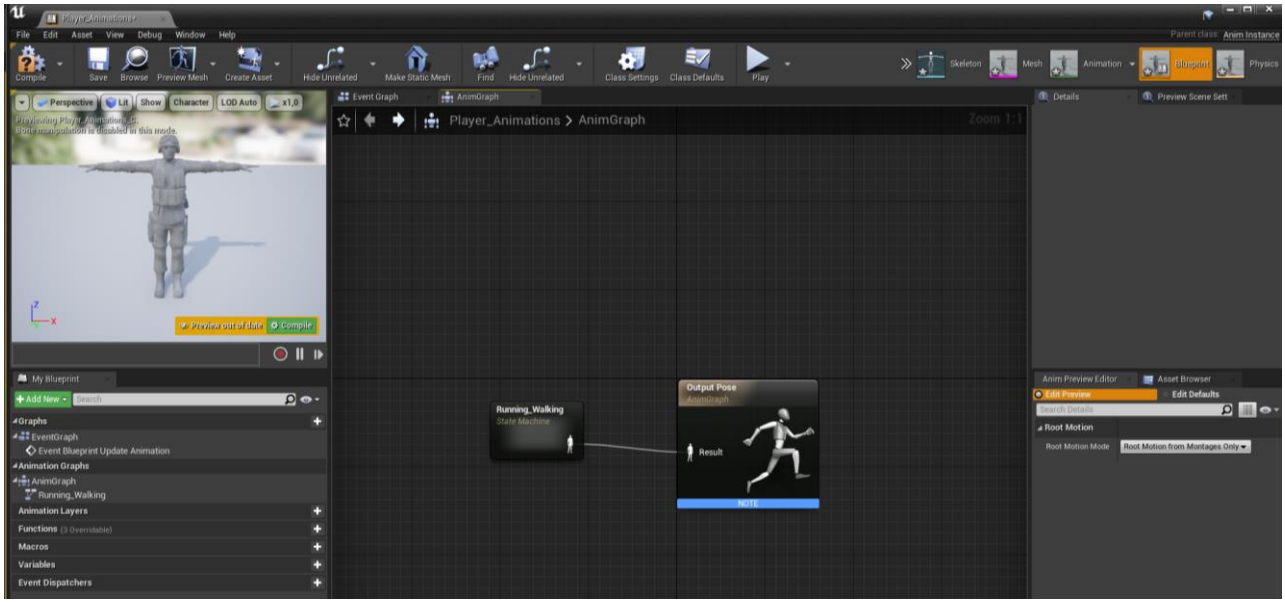


Рисунок 3.20 – Редактор Animation Blueprint

Маємо два стани: Idle та Run_Walk (див. рис. 3.21).

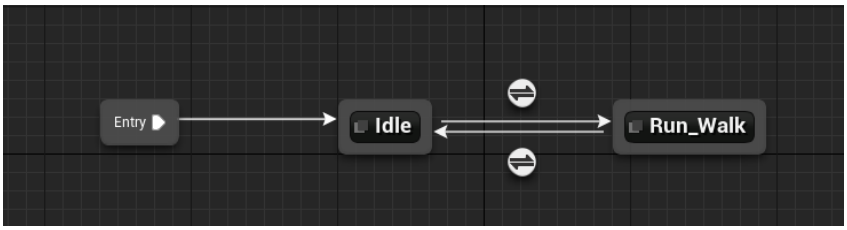


Рисунок 3.21 – State Machine

У стані Idle програватиметься анімація людини, яка стоїть на місті (див. рис. 3.22).

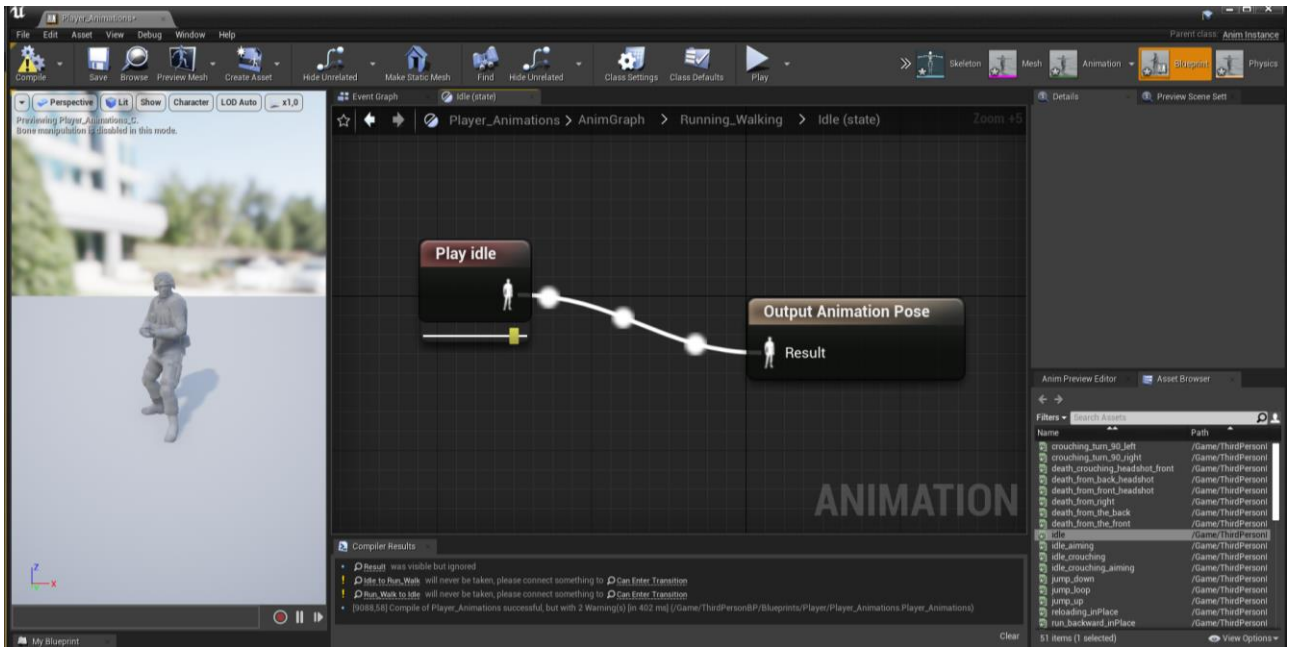


Рисунок 3.22 – Стан Idle

Перш, ніж додати анімацію ходьби, потрібно зрозуміти, куди і з якою швидкістю може рухатися наш персонаж. Створюємо ассет Blend Space. В його редакторі налаштовуємо анімації так, щоб вони змінювалися в залежності від швидкості та напряму руху гравця (див. рис. 3.23).

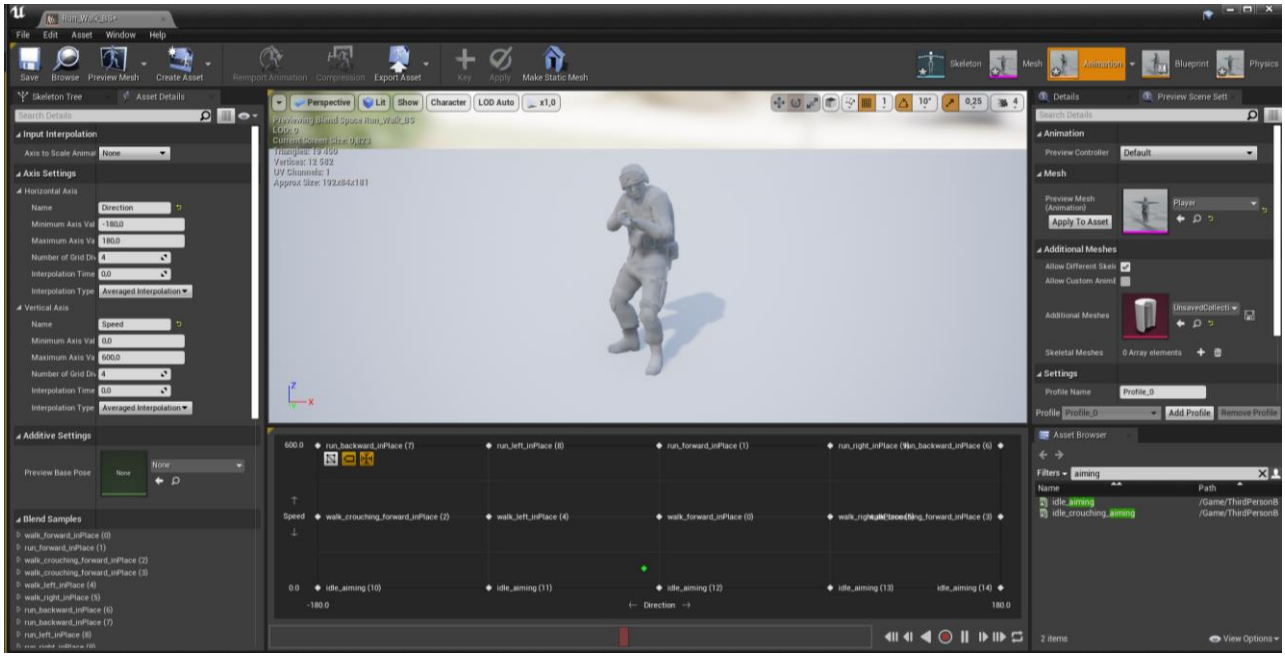


Рисунок 3.23 – Редактор Blend Space

Повертаємося в редактор створеного раніше кінцевого автомату. До стану Run_Walk додаємо створений Blend Space. В ньому треба вказувати напрям та швидкість руху персонажу. Ці дані необхідно буде розраховувати, тому створюємо дві змінні, в які будемо записувати всі необхідні дані (див. рис. 3.24).

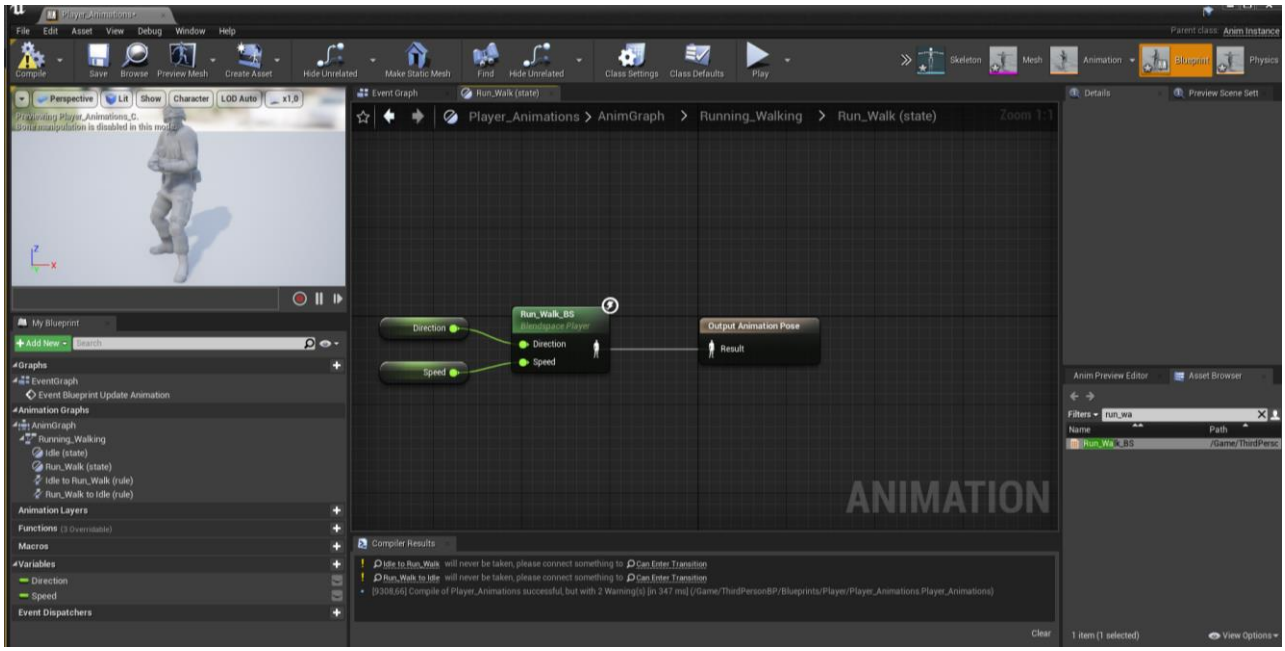


Рисунок 3.24 – Стан Run_Walk

Переходимо до вікна Event Graph та реалізуємо скрипт, за допомогою якого буде встановлюватися і напрям, і швидкість персонажа (див. рис. 3.25).

.

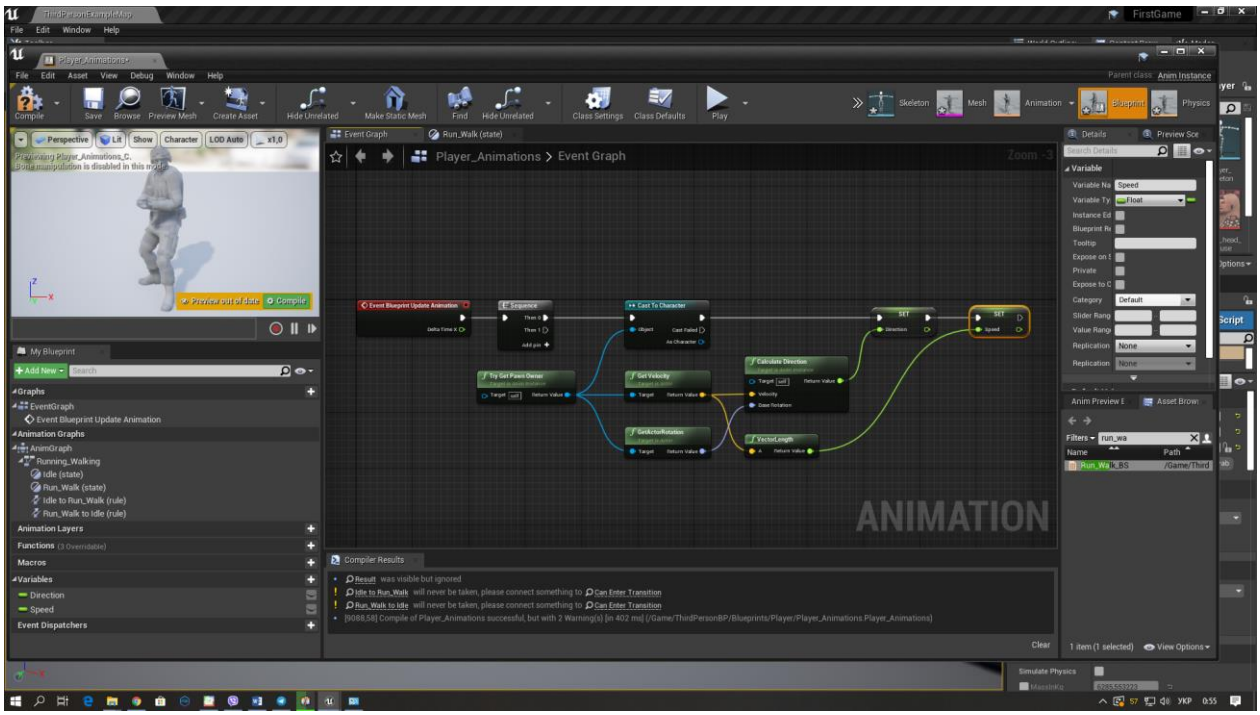


Рисунок 3.25 – Скрипт встановлення напряму і швидкості персонажа

Тепер необхідно вказати, коли саме буде здійснюватися перехід між станами стояння на місці та бігу. Налаштуємо все так, що якщо швидкість гравця стає більшою за десять, то запускається анімація бігу (див. рис. 3.26), а якщо стає меншою за десять, то запускається анімація стояння на місті (див. рис. 3.27).

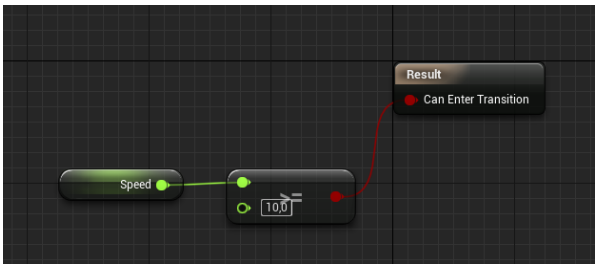


Рисунок 3.26 – Правило, при якому запускається анімація бігу

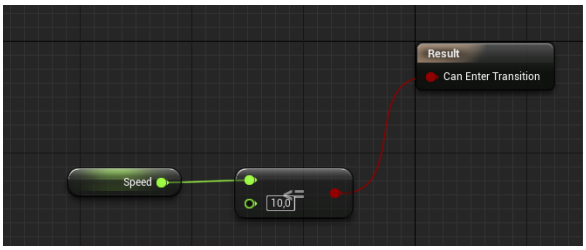


Рисунок 3.27 – Правило, при якому запускається анімація стояння на місці.

Останнім кроком замінюємо стандартного персонажа на нашого з потрібною нам анімацією (див. рис. 3.28). Після запуску гри можна побачити, що руки персонажа розташовані саме так, як треба, готові тримати зброю (див. рис. 3.29).

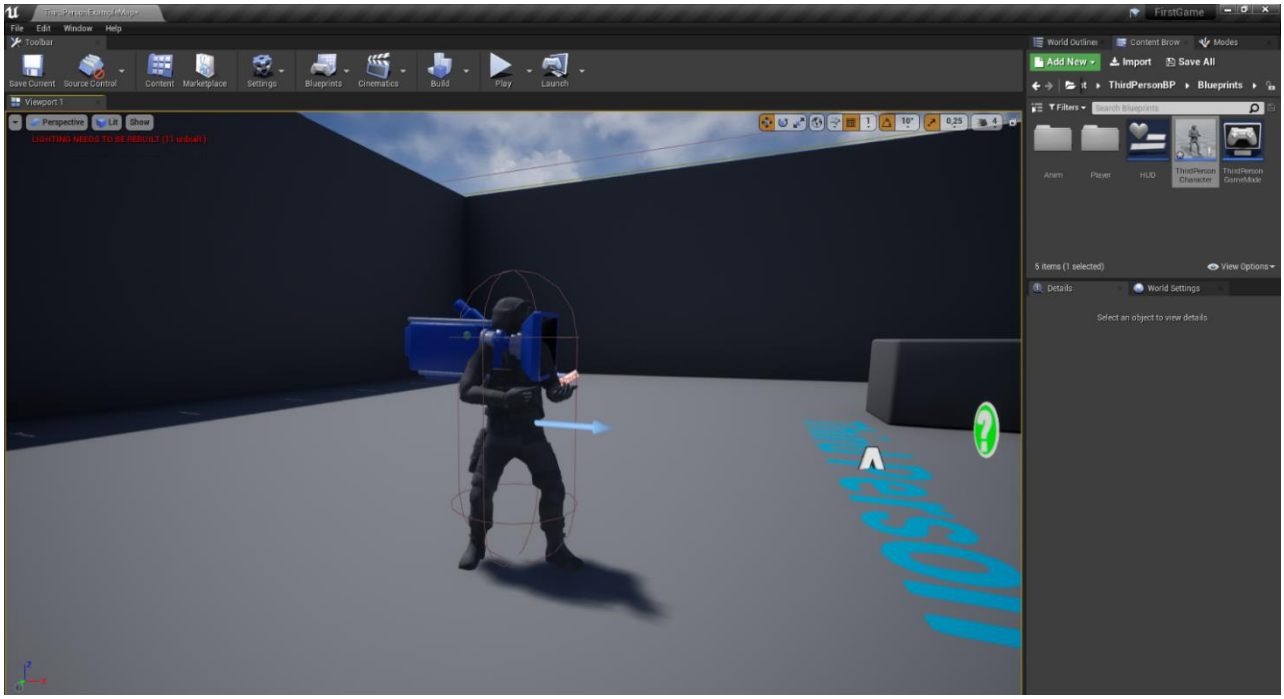


Рисунок 3.28 – Заміна стандартного персонажа

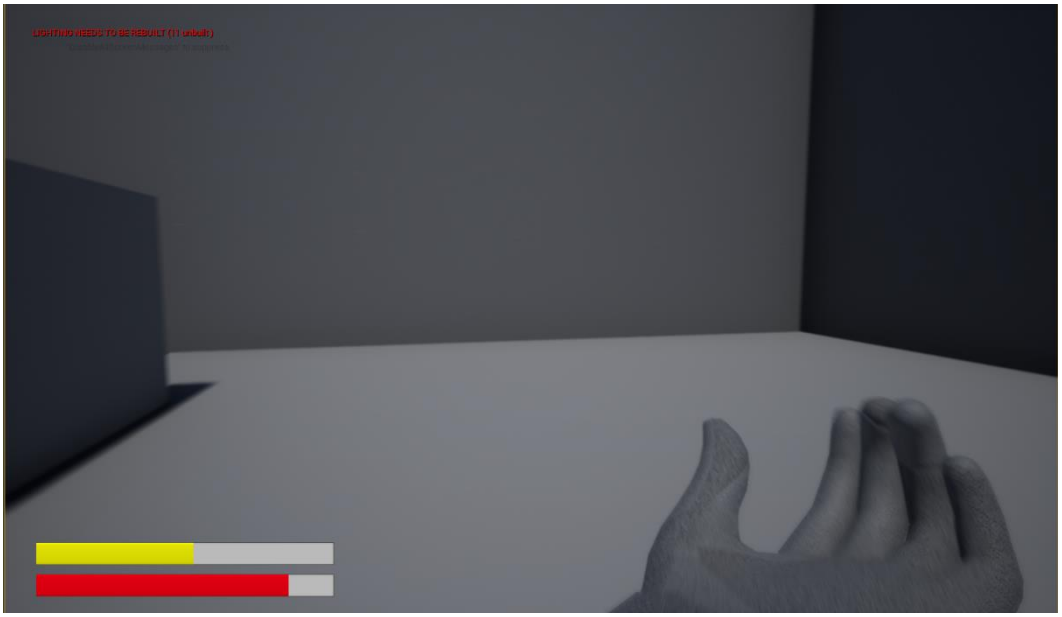


Рисунок 3.29 – Вид від першої особи

3.7. Додавання зброї до гравця

У шутері в гравця обов’язково повинна бути зброя. Видів зброї у світі дуже багато. Було вирішено зупинитися на автоматі. Серед готових об’єктів у відкритому доступі була знайдена модель автомата АК-47. Її було завантажено та додано до проекту. Тепер залишилося прив’язати автомат до ігрового персонажа.

Спочатку створюємо клас Gun, в якому основним елементом буде SkeletalMesh у вигляді автомату (див. рис. 3.30).

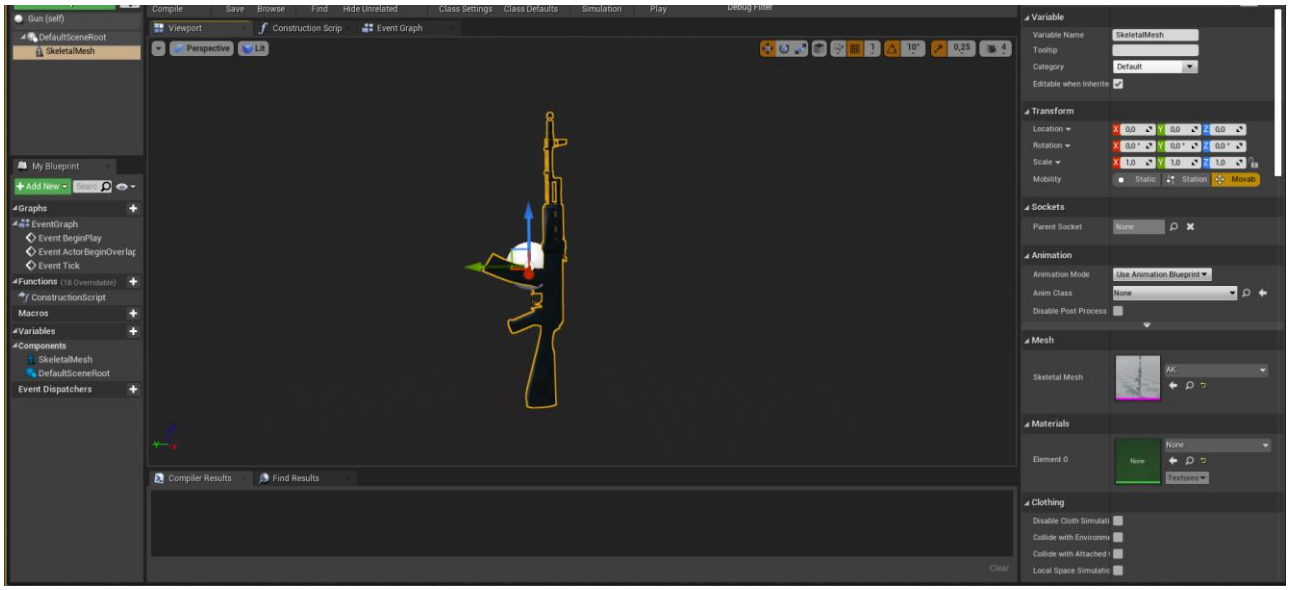


Рисунок 3.30 – Клас Gun

Далі треба за допомогою редактора скелету гравця прив’язати автомат до правої руки ігрового персонажу (див. рис. 3.31).

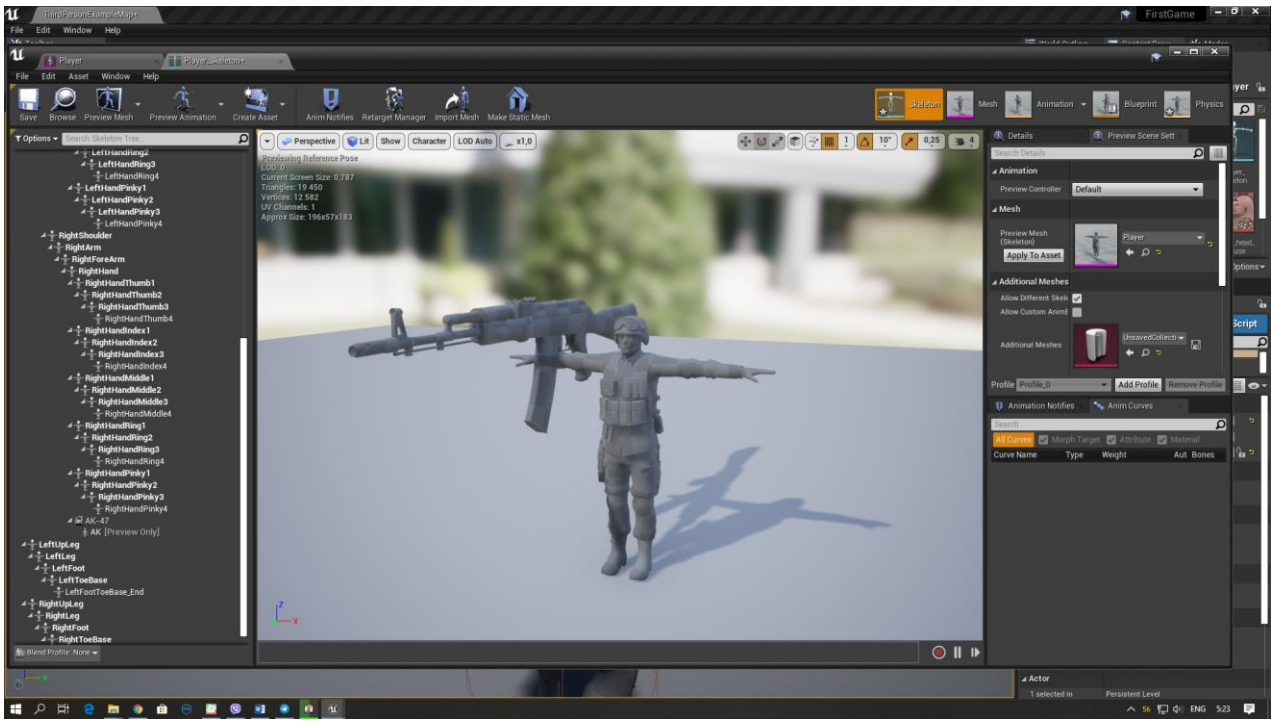


Рисунок 3.30 – Прив’язка автомату до руки персонажа

Як можна помітити, потребується налаштувати положення автомату. Спочатку треба активувати анімацію Idle_aiming у гравця, а потім за допомогою переміщень, поворотів та зменшення розмірів покласти автомат в руки персонажа (див. рис. 3.31).

Рисунок 3.31 – Налаштування положення автомату

Щоб після запуску гри гравцю давався автомат, треба доповнити блупринт ThirdPersonCharacter декількома скриптами (див. рис. 3.32). Потім перевіряємо, що все працює, як треба (див. рис. 3.33).

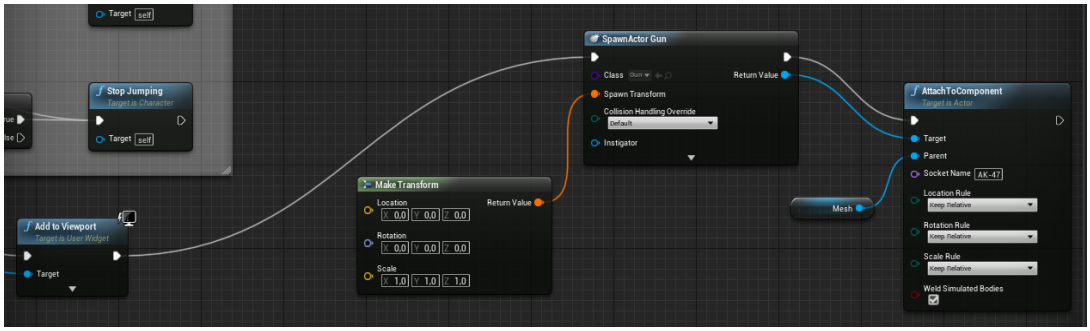


Рисунок 3.32 – Додання скриптів

Рисунок 3.33 – Перевірка роботи

3.8. Стрільба зі зброї

Спочатку треба створити новий клас, назвемо його Bullet. Основним компонентом цього класу буде меш Shape_NarrowCapsule. Це буде куля, якою зможе стріляти гравець за допомогою автомата. Далі додаємо компонент ProjectileMovement та налаштовуємо для кулі початкову та максимальну швидкості, а також вплив гравітації на неї (див. рис. 3.34).

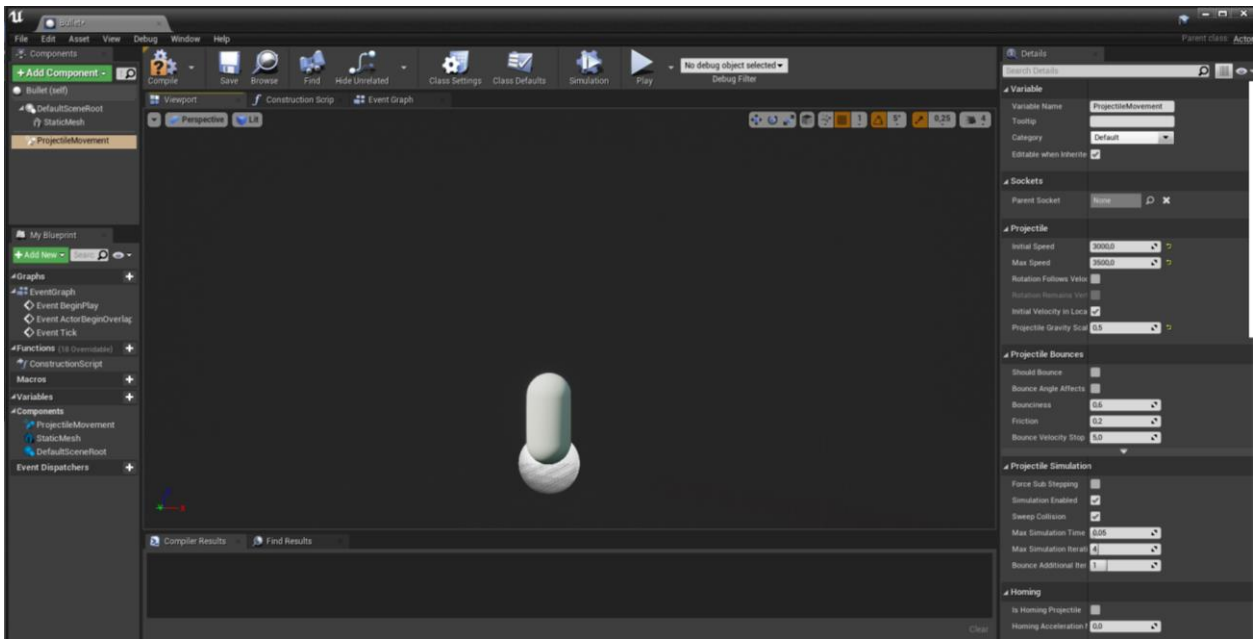


Рисунок 3.34 – Клас Bullet

Кулі під час стрільби повинні якби вилітати з дула автомата. Щоб реалізувати подібне, спочатку потрібно явно задати точку, в якій будуть з’являтися кулі. Для цього відкриваємо редактор скелету нашої зброї, додаємо так званий сокет та переміщаємо його в кінець дула (див. рис. 3.35).

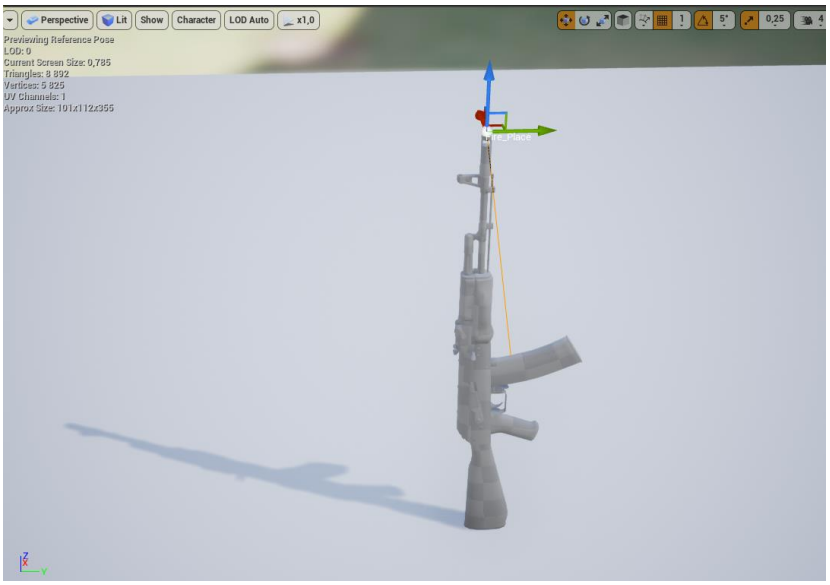


Рисунок 3.35 – Задання точки вильоту куль

Тепер відкриваємо блупринт клас Gun та створюємо нову подію, під час якої кулі будуть вилітати з дула автомата (див. рис. 3.36).

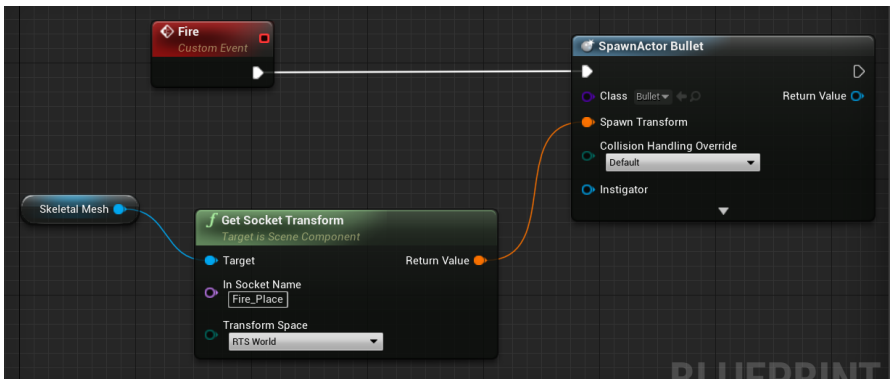


Рисунок 3.36 – Подія вильоту куль з автомату

Коли подія створена, потрібно вказати відстеження кнопок, при активації яких подія повинна відбуватися. Робиться це через налаштування Input в налаштуваннях проекту. Вказуємо, що подія Fire буде відбудеться при натисканні лівої кнопки миші (див. рис. 3.37).

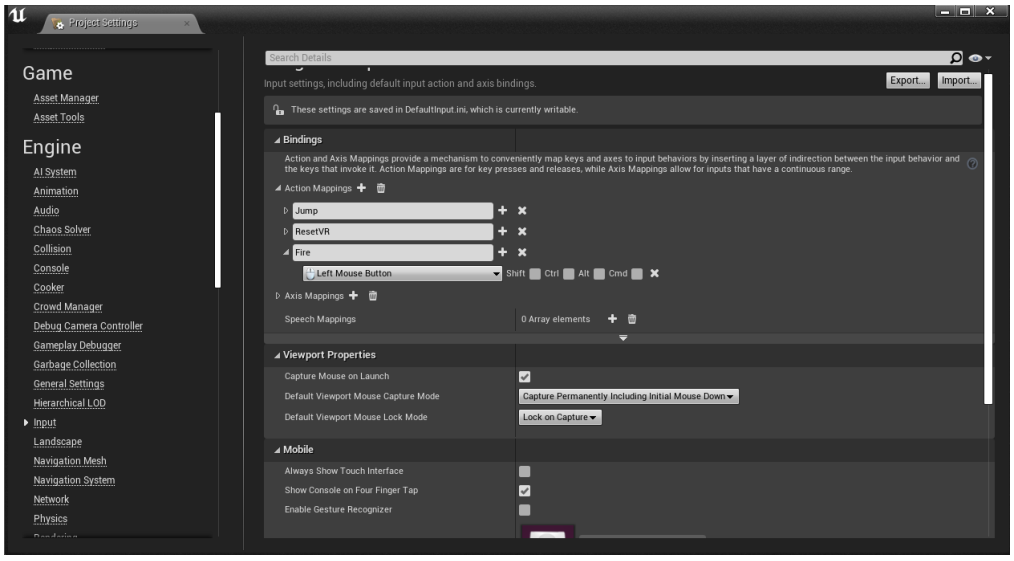


Рисунок 3.37 – Відстеження кнопок

Тепер потрібно в блупринті гравця реалізувати виклик події, при якій кулі будуть вистрелюватися з автомату. Спочатку треба створити посилання

на автомат, щоб була можливість брати у зброї потрібну нам подію (див. рис. 3.38).

А потім додаємо подію, яка при натисканні лівої кнопки миші запускає подію Fire у зброї гравця (див. рис. 3.39).

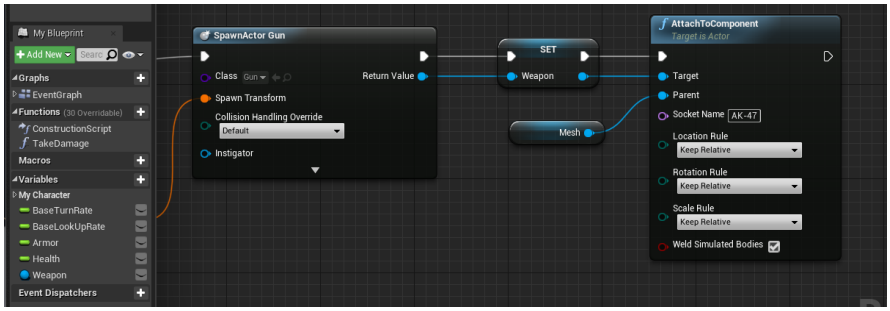


Рисунок 3.38 – Посилання на автомат

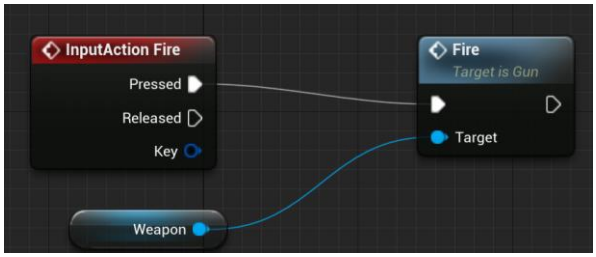


Рисунок 3.39 – Подія, яка запускає подію вильоту куль з автомату

Перед тим, як перевіряти правильність роботи стрільби, встановимо мале значення (200) швидкості польоту куль, щоб можна було з легкістю відстежити їх траєкторію. Під час перевірки було помічено, шо кулі летять не вперед, як потрібно, а вліво (див. рис. 3.40).

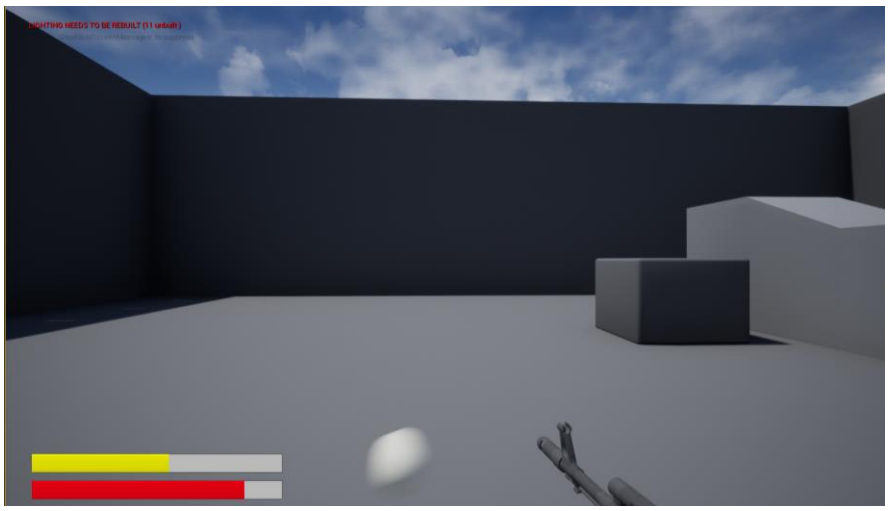


Рисунок 3.40 – Кулі летять вліво

Дана проблема була вирішена поворотом сокета Fire в редакторі скелету автомату. Тепер кулі летять вперед, але вилітають боком, не передньою своєю частиною (див. рис. 3.41).

Рисунок 3.41 – Кулі летять вперед, але боком

Дана проблема була вирішена поворотом кулі в редакторі блупринта Bullet (див. рис. 3.42).

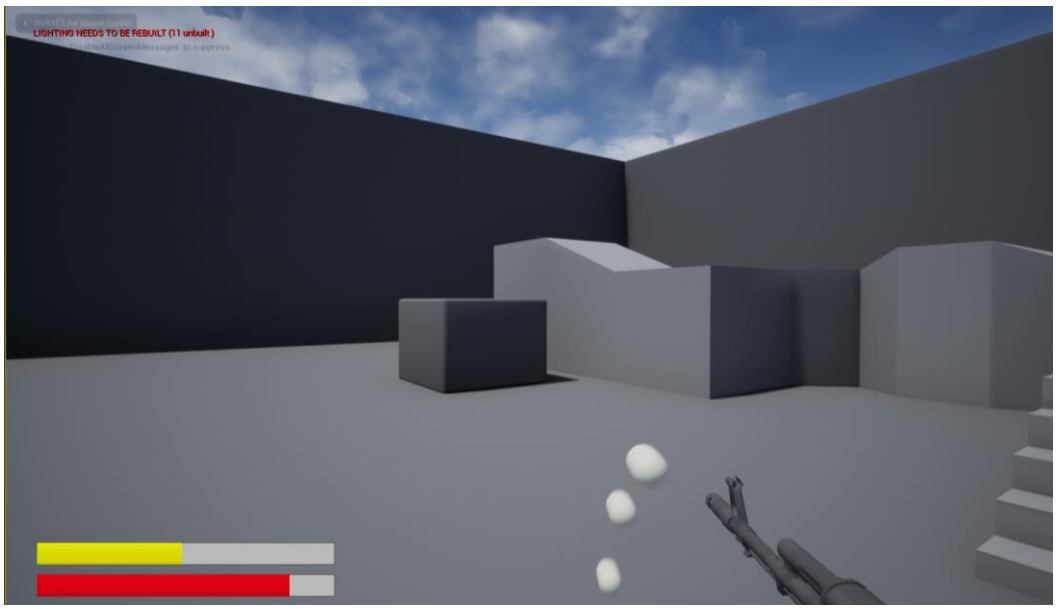


Рисунок 3.42 – Кулі летять як треба

Для більшого реалізму залишилося зменшити розміри куль та збільшити швидкість їх польоту (див. рис. 3.43).

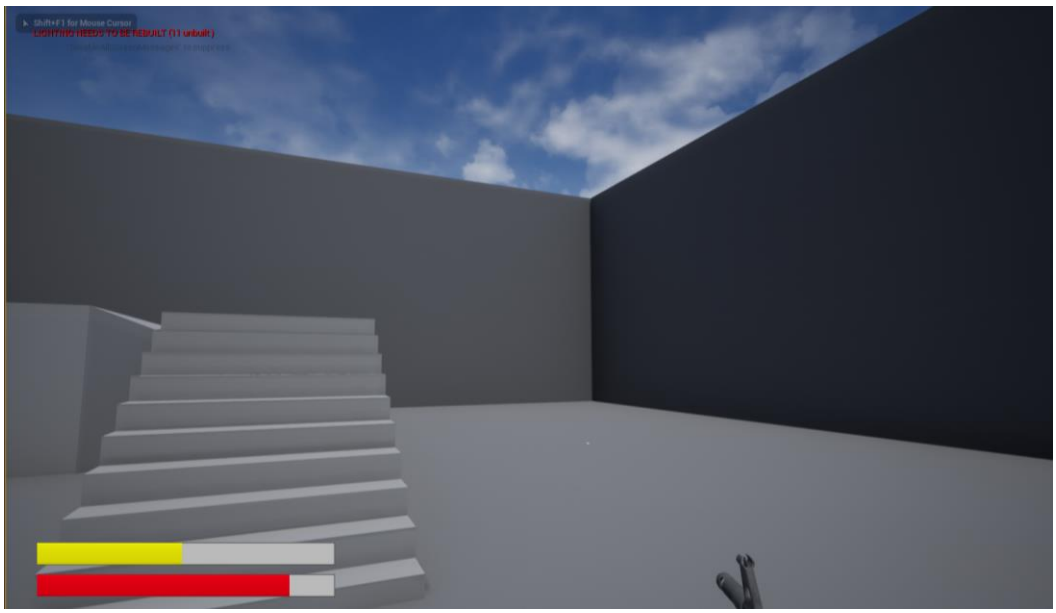


Рисунок 3.43 – Кінцевий варіант вильоту куль

3.9. Додавання ворогів та знищення їх

Головний ігровий персонаж повністю готовий. Тепер потрібно додати йому ворогів. Створюємо блупринт клас типу Character під назвою Enemy_AI

зі стандартним видом персонажа, зі стандартними анімаціями та змінною Health (див. рис. 3.44).

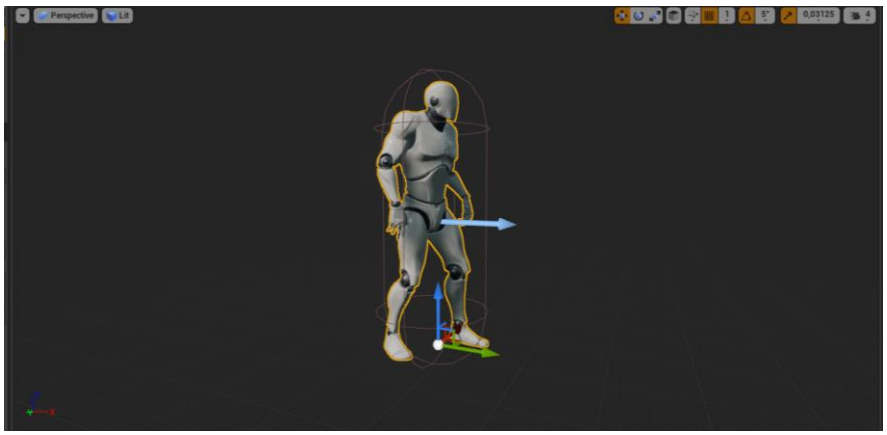


Рисунок 3.44 – Enemy_AI

За допомогою візуального програмування Blueprints налаштовуємо персонажа так, щоб при кожному попаданні кулі в нього віднімалося життя (див. рис. 3.45). Якщо рівень життя дійде до нуля, то ворог зникне (помре).

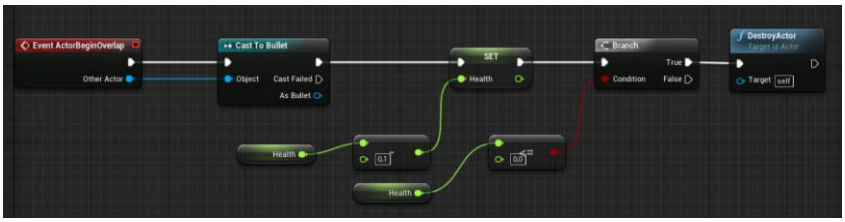


Рисунок 3.45 – Скрипт віднімання життя у ворогів

Перевірка роботи скрипта пройдена успішно (див. рис. 3.46, 3.47). Виявилося, що для вбивства ворога достатньо 10 куль.

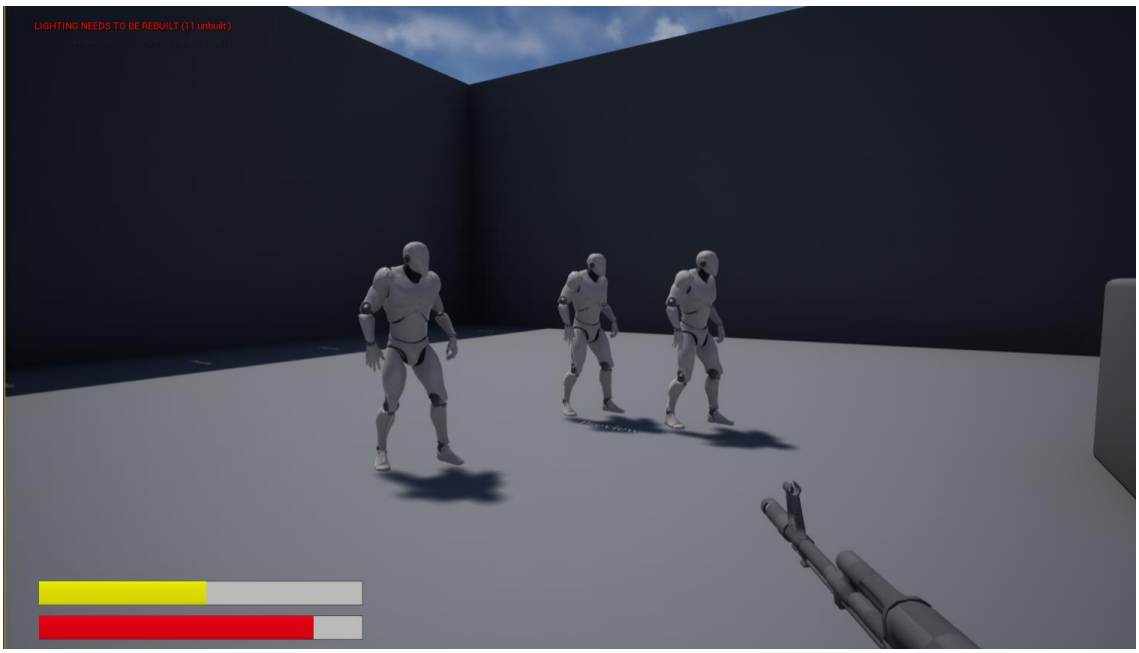


Рисунок 3.46 – Перед гравцем три вороги

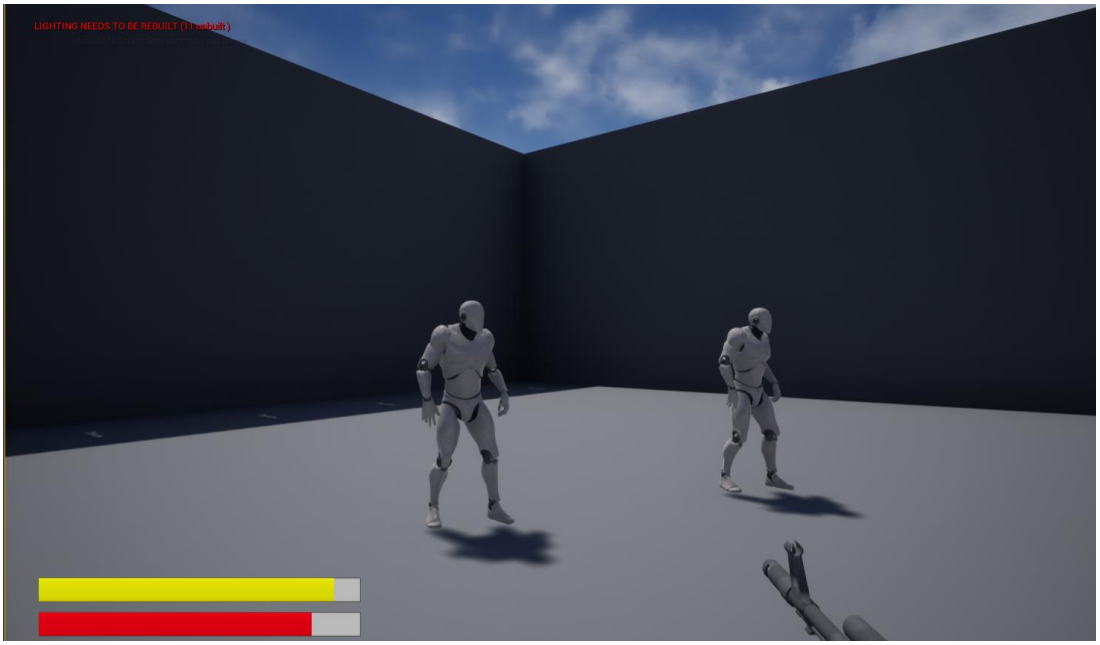


Рисунок 3.47 – Гравець вбив одного ворога, випустив в нього 10 куль

3.10. Урон від ворогів

Уявімо, що наші вороги – зомбі, які можуть тільки бити руками. Отже, урон по гравцю повинен наноситися тоді, коли він стоїть поруч з ворогом. Щоб це реалізувати, створюємо блупринт з назвою Pain та додаємо до нього компонент Capsule Collision.

Цей компонент – область у формі капсули. Треба його налаштувати так, щоб гравець, зайшовши в капсулу, отримував урон. В нашому випадку, коли гравець зайде в капсулу, то запуститься функція TakeDamage, яка була написана раніше. Реалізуємо все це за допомогою візуального програмування у редакторі Net Graph (див. рис. 3.48).

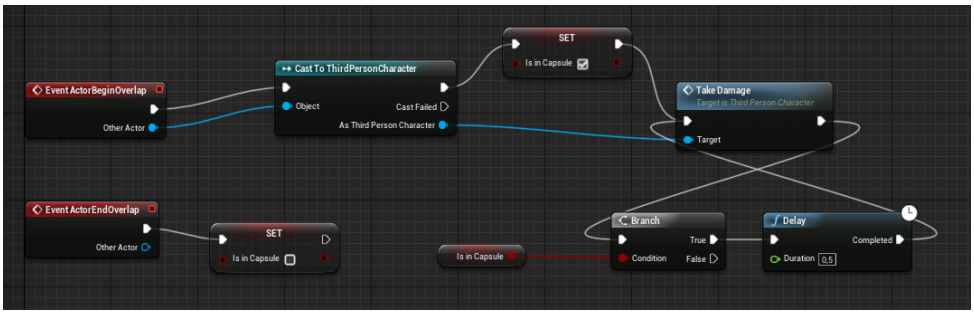


Рисунок 3.48 – Скрипт для капсули, що наносить урон гравцю

Далі потрібно прив’язати клас Pain до класу Enemy_AI. Тепер навколо кожного ворога буде зона, яка буде наносити урон гравцю, імітуючи ближній бій (див. рис. 3.49).

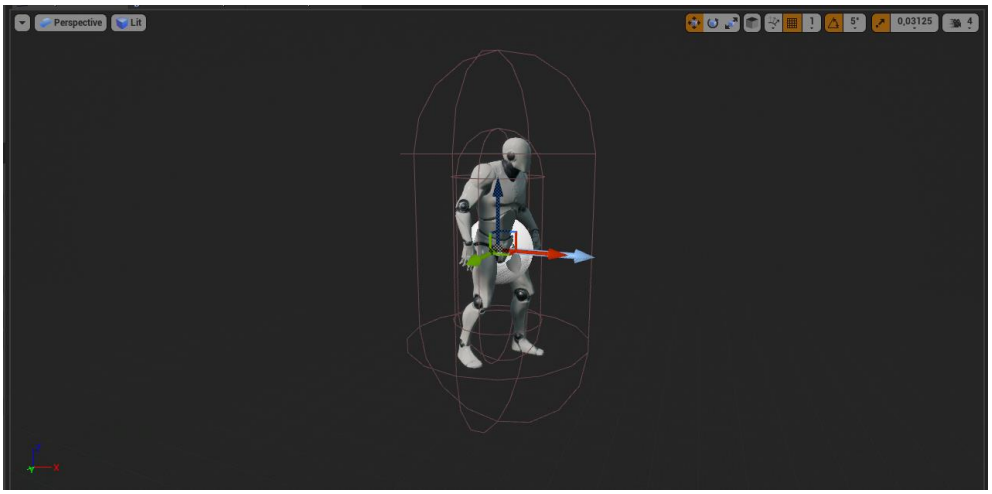


Рисунок 3.49 – Ворог з зоною навколо себе, яка нанесе урон гравцю

3.11. Штучний інтелект

Щоб вороги могли слідувати за гравцем, їм треба розповісти, де можна ходити, а де заборонено. Для цього з панелі Modes додаємо на рівень Nav Mesh Bounds Volume. Розтягуємо його на весь рівень, щоб персонажі зі

штучним інтелектом могли ходити по всьому рівню. Зелений колір показує, де саме вони можуть ходити (див. рис. 3.50) [6].

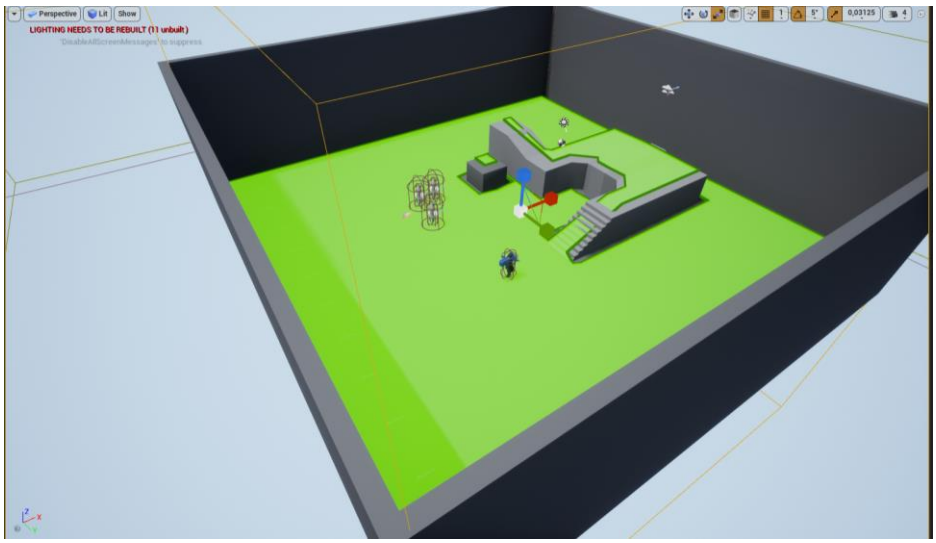


Рисунок 3.50 – Nav Mesh Bounds Volume

Додаємо до блупринта Enemy_AI компонент Pawn Sensing. Він дозволить персонажу зі штучним інтелектом бачити гравця, і як тільки він його помітить, цей момент можна відстежити та дати команду штучному інтелекту, наприклад команду наздогнати гравця.

В налаштуваннях Pawn Sensing ставимо кут огляду на 180 градусів (див. рис. 3.51). Це зроблено для того, щоб персонажі постійно відстежували гравця.

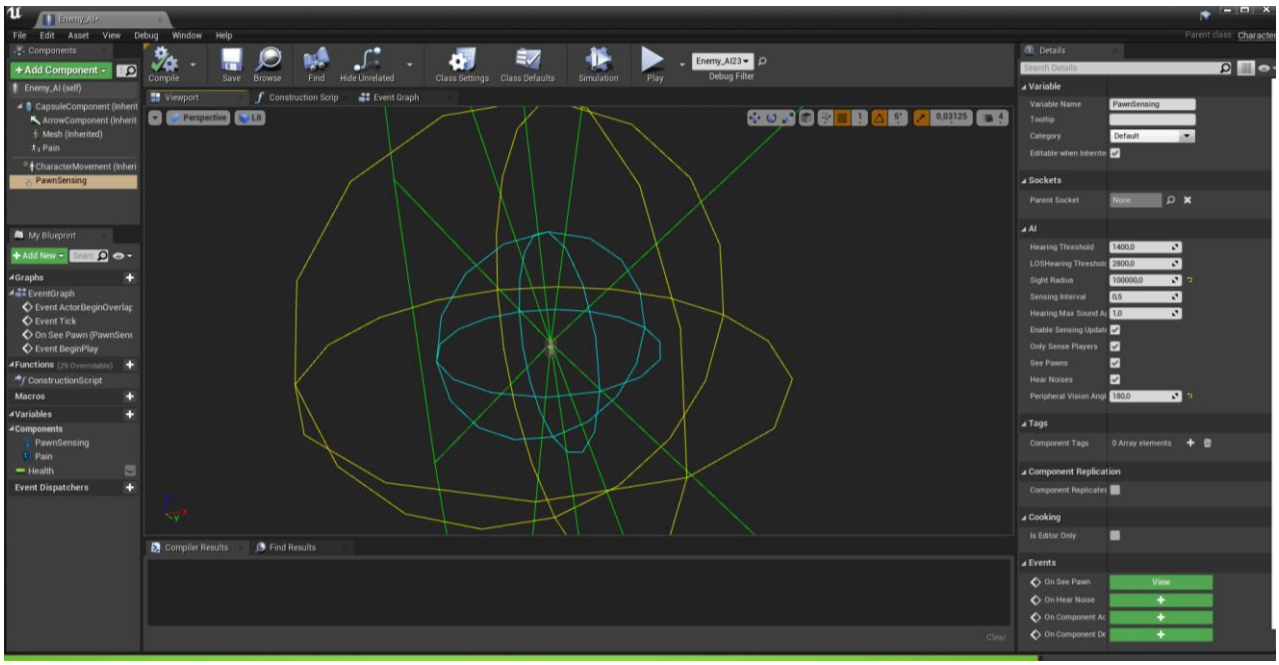


Рисунок 3.51 – Pawn Sensing

Тепер налаштовуємо подію, яка відбудеться, якщо гравець увійде в поле зору штучного інтелекту. Нехай в такому випадку персонаж зі штучним інтелектом почне бігти за гравцем, щоб максимально зблизитися з ним. Також потрібно підібрати для ворогів таку швидкість, щоб максимальна можлива швидкість гравця була більшою (див. рис. 3.52).

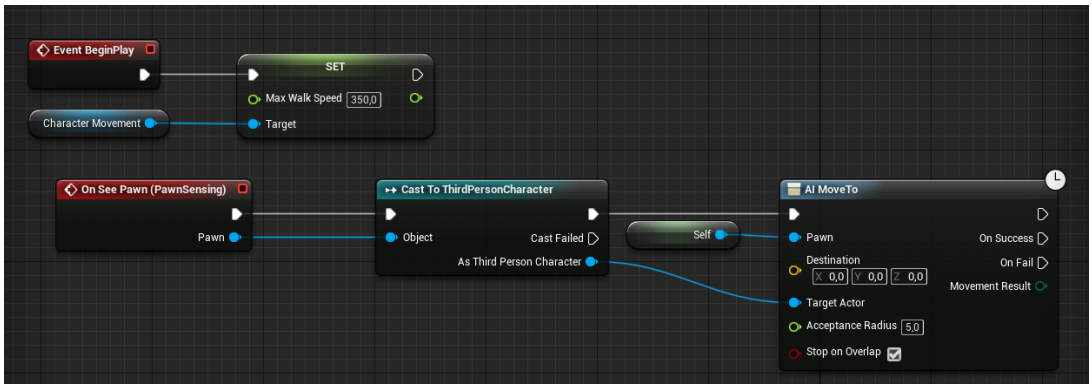


Рисунок 3.52 – Скрипт, який змушує ворогів постійно бігати за гравцем

Компілюємо та запускаємо гру. Все працює так, як і потрібно, вороги біжать за гравцем і якщо наздоганяють його, то наносять урон.

ВИСНОВКИ ДО РОЗДІЛУ 3

Успішно було розроблено основні частини геймплею комп’ютерної гри у жанрі шутер, її механіки. Для гравця був створений ігровий персонаж та автомат, з якого він може стріляти. Цьому персонажу було надано шкалу обладунків та шкалу життя. Також були створені вороги зі штучним інтелектом, які постійно переслідують та хочуть вбити гравця, зменшуючи його рівні обладунків та життя, якщо підходять близько до нього. У свою чергу, гравець може вбити ворога стріляючи з автомату.

РОЗДІЛ 4

4. ДЕМОНСТРАЦІЯ ВІДЕОГРИ

При запуску гри гравець з'являється у великій кімнаті в оточенні величезної кількості досить живучих ворогів, які відразу ж хочуть його вбити (див. рис. 4.1). Все, що вони можуть - це бігати за гравцем (див. рис. 4.2) і наносити йому урон, якщо зможуть наздогнати (див. рис. 4.3, 4.4). Гравець, в свою чергу, може бігати та вбивати ворогів стріляючи по ним з автомата (див. рис. 4.6). Коли гравець бігає від ворогів, а вони його не наздогнали, то обладунки гравця повільно відновлюються (див. рис. 4.5). Задача гравця – вижити, вбивши всіх ворогів, витративши не всю шкалу свого життя (див. рис. 4.7).

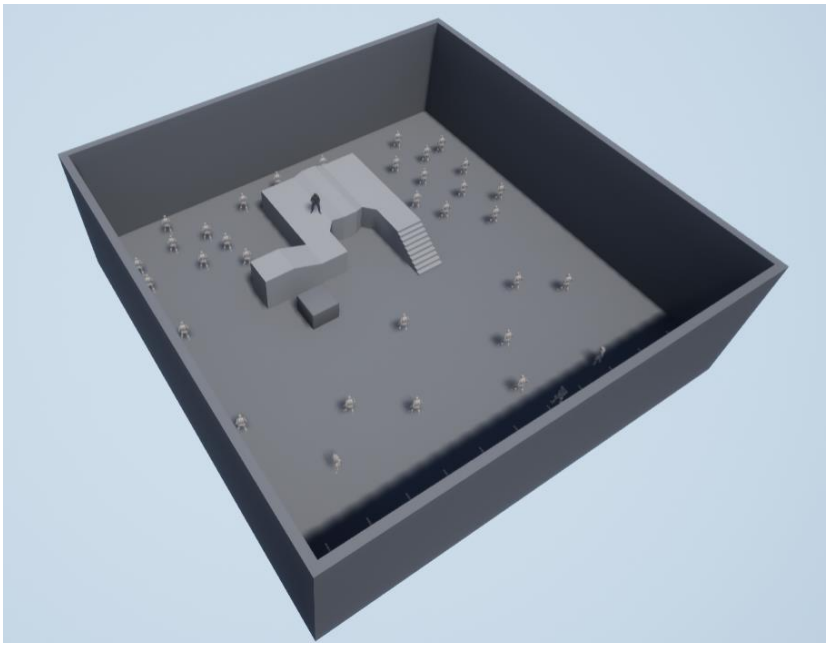


Рисунок 4.1 – Ігровий рівень в началі гри, вид згору

Рисунок 4.2 – Початок боротьби гравця з ворогами

Рисунок 4.3 – Вороги наздогнали гравця та почали наносити урон

Рисунок 4.4 – В гравця зламалися обладунки та почало закінчуватися життя

Рисунок 4.5 – Гравцю вдалося втекти, обладунки почали відновлятися

Рисунок 4.6 – Гравець вбив більшу кількість ворогів

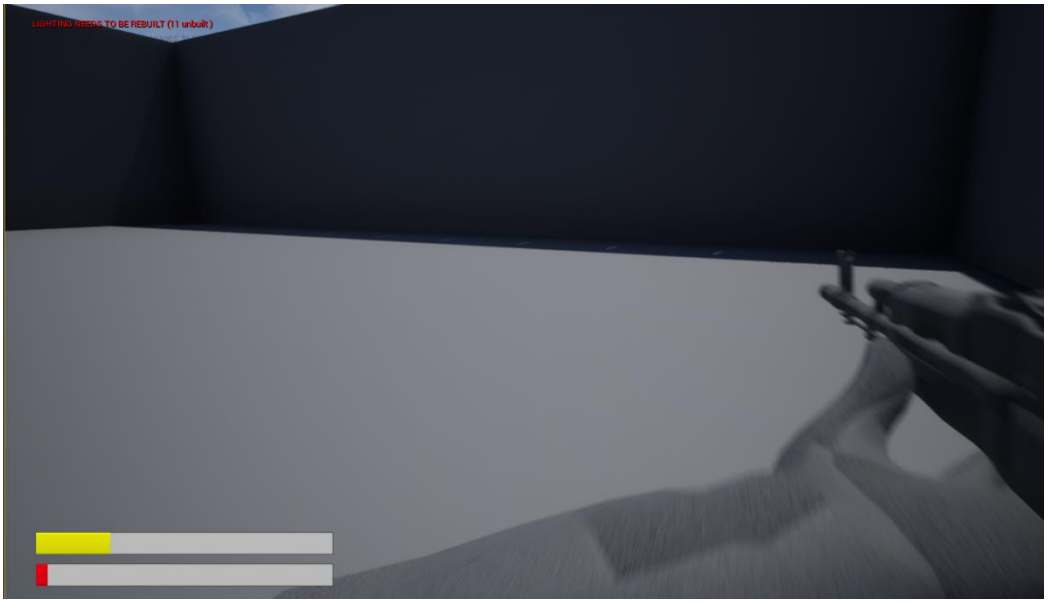


Рисунок 4.7 – Гравцю вдалося вбити всіх ворогів та вижити

ВИСНОВКИ ДО РОЗДІЛУ 4

Компьютерна гра вийшла цікавою та доволі складною. Реалізовані лише основні механіки, немає красивих текстур, крутих анімацій та звуків. Але цю гру спокійно можна використовувати як основу під час розробки крутого зомбі-шутера, оскільки найголовніше вже реалізовано – гравець зі зброєю та розумний штучний інтелект.

ВИСНОВКИ

Під час аналізу доступних джерел було проведено дослідження поняття комп'ютерна гра, під час якого була проведена класифікація ігор за 4 критеріями.

Додатково був складений алгоритм розробки відеоігор. Були проаналізовані популярні засоби розробки. В ході аналізу, було проведено їх порівняння і обрані найбільш актуальні засоби розробки для початкових розробників. Вибір пріоритетних засобів розробки проходив за двома критеріями: доступність і функціональність.

При аналізі існуючих розробок, було проведено їх порівняння і виділені їхні переваги і недоліки. В ході аналізу стало ясно, що при розробці комп'ютерної гри з простою ігровою механікою, варто звернути увагу на додаткові елементи гри, такі як сюжет і графічне оформлення. Це потрібно для того, що б утримати потенційного гравця і продовжити життєвий цикл розробки.

Після вибору засобів розробки було розпочато вивчення Unreal Engine 4, а також розробка самого проекту. В ході розробки був вивчений ігровий рушій Unreal Engine 4, візуальне програмування за допомогою Blueprints. Були придбані необхідні знання та вміння, а саме:

- створення рівнів;
- створення і написання скриптів;
- налаштування об'єктів;
- створення штучного інтелекту.

Освоєння середовища розробки Unreal Engine 4 несе не маловажний характер, так як в сучасному світі індустрія розробки ігор все сильніше поширюється в нашому суспільстві. Ігри перестали бути лише предметом для розваг, тепер використовуються і в інших областях, наприклад, в науці або в навчанні користувачів. Тому розвиток в даному напрямку можна вважати одним з найбільш важливих у сучасному суспільстві.

ПЕРЕЛІК ПОСИЛАНЬ

1. Unreal Engine Game Development Blueprints, 2015. – 352 с.
2. Carnall B. Unreal Engine 4.X By Example / Benjamin Carnall., 2016. – 506 с.
3. Куксон А. Разработка игр на Unreal Engine 4 за 24 часа / А. Куксон, Р. Даулінгсока, К. Крамплер., 2019. – 528 с. PV S. Unreal Engine 4 Game Development Essentials / Satheesh PV., 2016. – 266 с.
4. Valcasara N. Unreal Engine Game Development Blueprints / Nicola Valcasara., 2015. – 352 с.
5. Lee J. Learning Unreal Engine Game Development / Joanna Lee., 2016. – 274 с.
6. L. Newton P. Unreal Engine 4 AI Programming Essentials / Peter L. Newton., 2016. – 188 с.
7. A.Moniem M. Mastering Unreal Engine 4.X / Muhammad A.Moniem., 2016. – 384 с.
8. Sewell B. Blueprints Visual Scripting for Unreal Engine / Brenden Sewell., 2015. – 188 с.
9. Tavakkoli A. Game Development and Simulation with Unreal Technology / Alireza Tavakkoli., 2015. – 742 с.
10. Lee J. Unreal Engine: Game Development from A to Z / J. Lee, J. P. Doran, N. Misra., 2016. – 837 с.
11. Edmonds M. Mastering Game Development with Unreal Engine 4 / Matt Edmonds., 2018. – 356 с.
12. Unreal Engine 4 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unrealengine.com/en-US/index.html.
13. Туториал по Unreal Engine [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://habr.com/ru/post/344394/.

14. Навигация Вьюпорта в Unreal Engine 4 [Електронний ресурс] – Режим доступу до ресурсу: https://www.native-game.com/ue4-docs/ue4-manual/navigacija-vjuporta-v-unreal-engine-4/.
15. Как Импортировать Анимацию в Unreal Engine 4 [Електронний ресурс] – Режим доступу до ресурсу: https://www.native-game.com/ue4-docs/ue4-manual/kak-importirovat-animaciju-v-unreal-engine-4/.

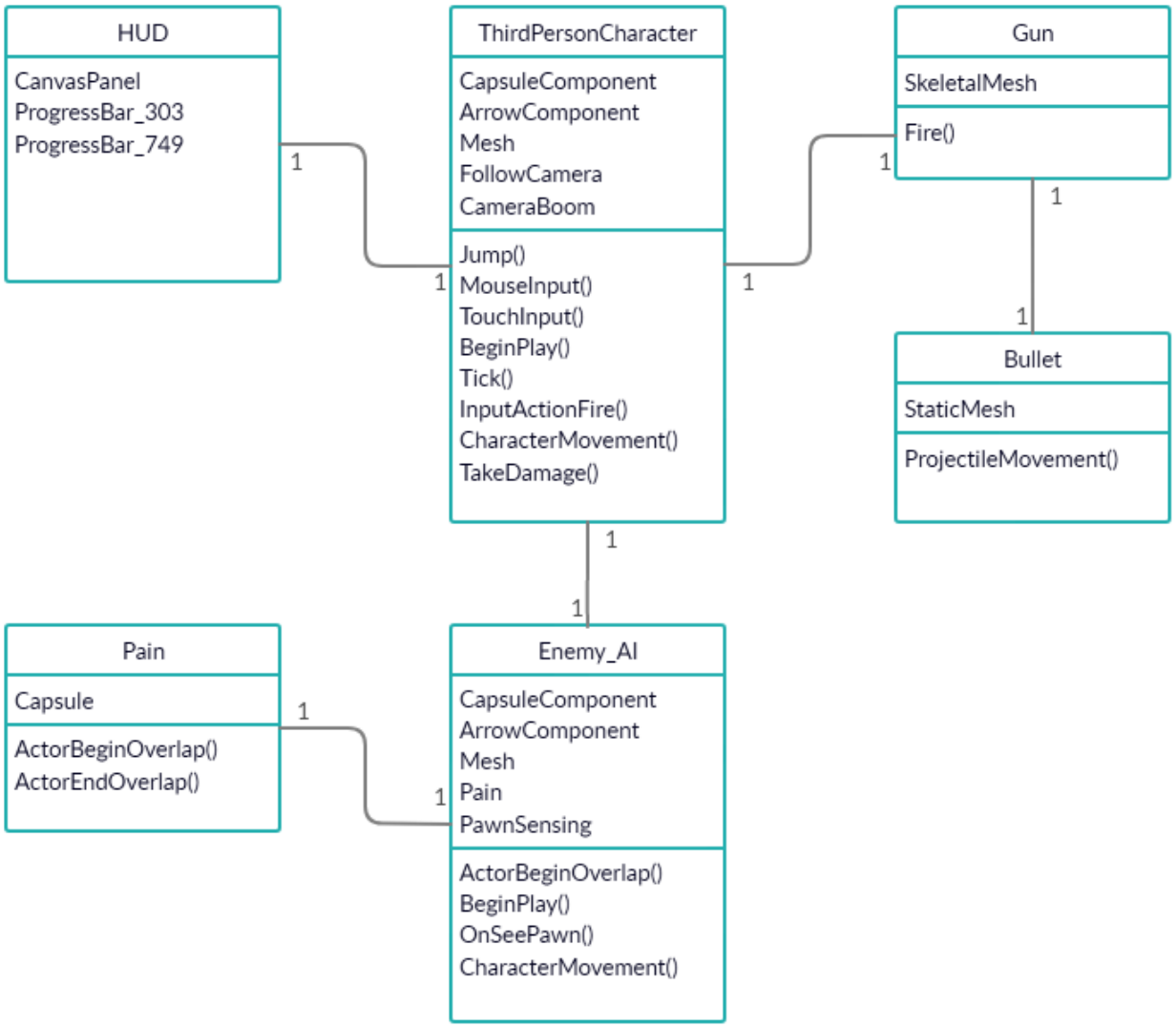
Додаток 1

до дипломного проєкту

на тему: «Комп'ютерна гра

на базі рушія Unreal Engine 4»

Київ – 2020 року



					ІАЛЦ. 467800.003 Д1		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Маслачков О.Ю.			Комп'ютерна гра на базі рушія Unreal Engine 4	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Сімоненко А.В.					
Н. контр.		Сімоненко В.П.			Функціональна схема	НТУУ "КПІ", ФІОТ, ІО-63	
Затверд.							

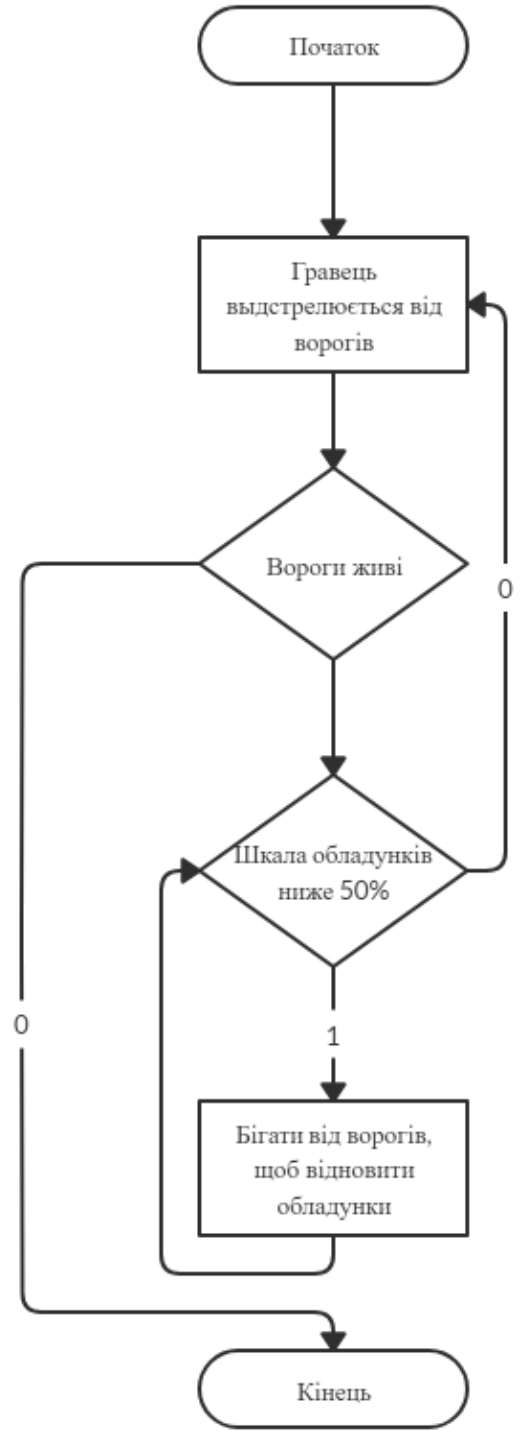
Додаток 2

до дипломного проєкту

на тему: «Комп'ютерна гра

на базі рушія Unreal Engine 4»

Київ – 2020 року



					ІАЛЦ. 467800.003 Д2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Маслачков О.Ю.			Комп'ютерна гра на базі рушія Unreal Engine 4	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Сімоненко А.В.					
Н. контр.		Сімоненко В.П.			Принципова схема	НТУУ "КПІ", ФІОТ, ІО-63	
Затверд.							

Додаток 3

до дипломного проєкту

на тему: «Комп'ютерна гра

на базі рушія Unreal Engine 4»

Київ – 2020 року

					ІАЛЦ. 467800.003 ДЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Маслачков О.Ю.			Комп'ютерна гра на базі рушія Unreal Engine 4	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Сімоненко А.В.					
Н. контр.		Сімоненко В.П.			Структурна схема	НТУУ "КПІ", ФІОТ, ІО-63	
Затверд.							